VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Generátor výstupních sestav - rozšíření
Generator

2012 Martin Dočkal

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Dočkal**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Generátor výstupních sestav - rozšíření
Generator

Zásady pro vypracování:

Diplomant má za úkol rozšířit a vylepšit současné řešení generátoru výstupních sestav (viz Seznam doporučené literatury).
1. Prostudujte současná řešení generátoru výstupních sestav.
2. Prostudujte reportovací služby v MS SQL Serveru 2008, případně jiné generátory sestav.
3. Navrhněte rozšíření (funkce SQL, poddotazy, jiný formát výstupu, jiné vstupní SŘBD apod.) současného řešení generátoru výstupních sestav.
4. Vypracujte analýzu a návrh implementace rozšířeného generátoru výstupních sestav.
5. Navržený generátor implementujte a otestujte.
6. Napište programátorskou a uživatelskou příručku.
7. Navržený generátor porovnejte s reportovacími službami MS SQL Serveru 2008 i nástroji dalších výrobců.

Seznam doporučené odborné literatury:

BARTONÍČEK, Adam. Generátor výstupních sestav. Ostrava, 2010. 90 s. Diplomová práce. VŠB TUO, Fakulta elektrotechniky a informatiky.
LACKO, Luboslav. Business Inteligence v SQL Serveru 2008: Reportovací, analytické a další datové služby. Vydání první. Brno: Computer Press, a.s., 2009. Kapitola 7 Reportovací služby, s. 323-418. ISBN 978-80-251-2887-9.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Emilie Šeptáková**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovu práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2012, Martin Dočkal

Poděkování

Především bych chtěl poděkovat **Ing. Emilii Šeptákové**, vedoucímu této diplomové práce, která podpořila můj záměr zpracovat svou práci na základě rozšíření poznatků jiné níže uvedené diplomové práce a pravidelně mi byla v rámci konzultací k dispozici za účelem kontrol a dalšího vývoje této práce

Abstrakt

Práce se zabývá zpracováním dat z databázových zdrojů, získaných z různých informačních systémů, za účelem dalších analýz pro potřeby dalšího rozhodování. Cílem práce je navržení a implementace aplikace, jenž na základě vstupních dat a interakci obsluhujícího uživatele vygeneruje skript s danými filtry, a to i pro uživatele bez nutné znalosti a zkušeností z oblasti databázových technologií. Důvodem je stále rostoucí počet informačních systémů a snaha o zjednodušení práce s daty napříč různými datovými typy a zpřístupnit je i běžným uživatelům bez znalosti problematiky jak z oblasti databáze, tak jejich jednotlivými typy, kteří by dané sestavy za jiných okolností bez využití generátoru a jeho průvodce buď nebyli vytvořit uplně, případně by si vyžádalo značné úsilí z jejich strany, a to při každém novém budování takového výstupu. Práce staví na základech jiné diplomové práce shodného tématu, přičemž je zaměřena na rozšiřující prvky, a to jak v části teoretické, tak praktické.

Klíčová slova: Databázový systém, report, analýza, informační systém, výstupní sestava, generátor, skript.

Abstract

This thesis deals with data processing from database sources. These sources are obtained from different information systems in order to make another analysis for needs of following decision. The aim of this work is to design and implement application which based on input data and interaction of serving user, generates a script with existing filters. The user does not have to be familiar with database systems. The reason is still and still growing number of information systems and effort to make them more user - friendly. The result should be more friendly work with data, even over their wide span, even for non - professional users without knowledge of database systems and its subsystems. Otherwise the users would challenge a difficult task to work without generator and its guide and the result could be even unachievable or they would have to put a tremendous amount of energy while building such a new output. This diploma thesis is an extension of another diploma thesis with an identical topic, and is focused on expanding elements, both in theoretical and practical part.

Keywords: Database system, report, analysis, information system, output, generator, script.

Seznam použitých zkratek a symbolů

API	Application Programming Interface
ASP	Active Server Pages
BIDS	Business Intelligence Development Studio
BI	Business Inteligence
CS	C Sharp
CSS	Cascading Style Sheets
DML	Data Manipulation Language
DWH	Data Warehouse
GB	Giga Byte
HTTP	HyperText Transfer Protocol
HW	Harwdare
IDPL	Initial Developer's Public License
IT	Information Technology
JDBC	Java Database Connectivity
MOSS	MS Office SharePoint Server
MS	Microsoft
MVC	Model View Controller
ODBC	Open Database Connectivity
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
PDO	PHP Data Objects
PHP	Personal Home Page, Hypertext Preprocessor
RDL	Report Definition Language
ROM	Rendering Object Model
RS	Reporting Services
SP	Service Pack
SQL	Structured Query Language
SŘBD	Systém řízení báze dat
SSRS	SQL Server Reporting Services
SW	Software
TB	Tera Byte
WAL	Write ahead logging
WWW	World Wide Web
XHTML	eXtensible HyperText Markup Language
XML	Extensible Markup Language
.NET	aplikační platforma společnosti Microsoft

Obsah

Seznam Umístění na DVD

1 Úvod

Hlavní náplní této diplomové práce je vývoj aplikace generující skript PHP z rozličných typů databáze. Protože se jedná se o rozšíření práce pana Ing. Bartoníčka [1], je kladen větší důraz na rozšiřující segmenty, zatímco v původní práci je detailně popsán např. rozbor SQL dotazů či technologie .NET, na kterou se budu odkazovat, v této samotné práci ovšem zvlášť a znovu popisovat již nebudu, výjimkou budou tvořit některé odlišnosti či novinky (např. v technologii .NET ap.). Jednotlivé kapitoly této práce jsou vztaženy ze zadání, kde je rozebrán a popsán každý bod samostatně.

V následující kapitole je shrnutí zmíněné práce pana Ing. Bartoníčka. Popisuje hlavní body práce včetně hlavních znaků výsledné aplikace - generátoru.

V kapitole Reportovací služby je teoretický popis a rozbor základních pojmů z oblasti BI a datových skladů, aby měl čtenář lepší nadhled a představu, do jaké oblasti práce spadá. Dále je zde uveden vybraný výčet nástrojů ze současné praxe.

Kapitola rozšíření současného generátoru výstupních sestav obsahuje hlavní rozšíření původní práce s důrazem na výslednou aplikaci.

Protože aplikace spadá do kategorie informačních systémů, následuje analýza systému, zejména datová a funkční.

Mezi analýzou a samotnou implementací systému se nachází kategorie Návrh implementace, v níž je uvedeno implementační prostředí, instalace IS, popis výchozího prostředí apod.

Etapa implementace zaujala největší podíl, což je jeden z ukazatelů směrující na složitost a rozsáhlost aplikace. V textu této aplikace je pak uveden výsledek vygenerovaný již výsledným novým a rozšířeným generátorem.

Spolu s vyvinutým generátorem byla sepsána také uživatelská příručka popisující ovládání systému, z níž je uvedena ukázka také v tomto textu v kapitole zabývající se právě uživatelskou příručkou, kompletní znění je pak přiloženo formou přílohy na elektronickém médiu podobně, jako řada dalších dokumentů a souborů, jenž doprovázejí tuto diplomovou práci.

Programátorská dokumentace je rovněž přiložena na médiu, a to především formou dokumentace na základě komentářů ve zdrojových souborech a třídního diagramu. V textu této práce je pak uvedena také struktura výsledné aplikace.

Poslední dvě kapitoly se zabývají závěrečným shrnutím celé práce. První se zabývá srovnáním nového generátoru s původním i dalšími nástroji, v druhé, závěrečné, je pak výsledek práce začleněn do širšího kontextu, resp. s popisem základních vlastností a rozšíření, které byly realizovány s návrhem pro další rozšíření.

2 Současné řešení generátoru výstupních sestav

V původní práci jsou detailně popsány typy sestav s detailním popisem včetně ukázkových příkladů [1]. Výčet popisovaných typů:

- příkaz SELECT,
- výběr dat z jedné tabulky,
- výběr dat z více tabulek,
- výběr dat 1:N,
- výběr dat vazební tabulka,

přičemž hlavní důraz je kladen na prvně jmenovaný, tj. samotný příkaz SELECT.

Původní generátor výstupních sestav jako webová aplikace obsahuje následující výčet funkcí:

- test spojení,
- vytvoření spojení,
- přehled spojení,
- editace spojení,
- vymazání spojení,
- vytvoření spojení,
- přehled sestav,
- editace sestavy,
- vymazání sestavy,
- vytvoření uživatele,
- přehled uživatelů,
- editace uživatele,
- vymazání uživatele,
- přihlášení do aplikace,
- odhlášení z aplikace,
- změna hesla,
- obnova hesla.

Tyto body včetně základního zadání, tj. vstupu do systému, se pak staly základem pro můj rozšířený generátor, přičemž některé již realizovány nebyly (např. obnova hesla), některé byly přidány (např. typ spojení) , sloučeny (např. vytváření sestavy) či pozměněny.

3 Reportovací služby

Cílem této kapitoly je seznámení se se základními pojmy z oblasti reportovacích služeb.

3.1 Úvod

Report obecně chápeme jako nástroj, který ze vstupních dat, jakožto základní atomickou databázovou jednotkou, které mezi sebou jsou vzájemně v interakcích a tvoří tedy informace, jenž nejsou nebo nemusí být na první pohled zřejmé, nevhodně či vůbec vizuálně podány, přeměnou vytvoří nová, případně zvýrazní zmíněné souvislosti mezi nimi, to vše za účelem dalšího rozhodování v závislosti na typu reportu.

Popsaný proces takové přeměny většinou není devizí jediného nástroje, ale tvoří ucelenou skupinu nástrojů, které spolu většinou vzájemně spolupracují, resp. výstupem jednoho nástroje je obvykle vstupem druhého nástroje, které v tomto případně označujeme jako reportovací nástroje a řadíme je do kategorie tzv. Business Inteligence. Vytváření reportovacích sestav a prezentace jejich výsledků je většinou odpovědností analytiků.

3.2 Business Inteligence

Tímto pojmem, zavedeným již v 60. letech 20. století, rozumíme metodiky využívané v podnikání za účelem podpory rozhodování v závislosti na znalostech, sběru a vyhodnocování, jenž přemění data na informace a informace na poznatky a získání lepšího pochopení chování zkoumané oblasti, zejména ale pak řízení výkonnosti zkoumaného odvětví. Provádění rozhodování na základně výsledků analytických týmů a výsledků je většinou odpovědností manažerských rolí.

V roce 1989 definoval formálně tento pojem následující citací [2]:

Bussines Inteligence je množina konceptů a metodik, které zlepšují rozhodovací proces za použití metrik, nebo systémů založených na metrikách. Účelem procesu je konvertovat velké objemy dat na poznatky, které jsou potřebné pro koncové uživatele. Tyto poznatky potom můžeme efektivně použít například v procesu rozhodování a mohout tvořit velmi významnou konkurenční výhodu

Kompletní řešení Business Inteligence nám ve finální části zkoumané oblasti dokáže pomoci zodpovědět na následující otázky:

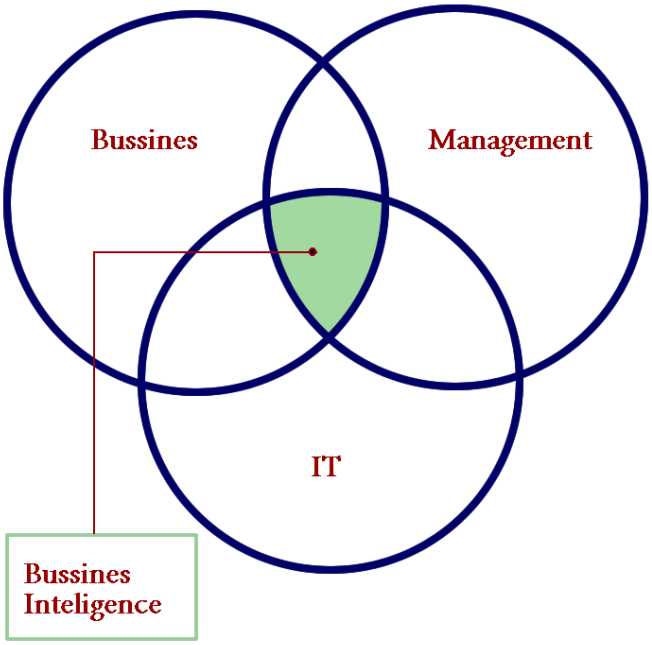
- Jak vypadá současný stav,
- proč se tak děje,
- předpovědět co se bude dít,
- jaké možnosti by měly postupně následovat,
- zvýšení efektivity spolupráce týmu.

Hlavní dnešní datové zdroje pro BI jsou čerpány primárně z datových skladů nebo případně z tzv. datových trhů *(data marts)*.

Zjednodušeně lze říct, že Business Inteligence je vzájemný průnik následujících fragmentů:

- Předmět zkoumaného odvětví či sféry (např. obchodu),
- jeho správa (management),
- informační technologie a jejich aplikování.

Grafický znázorněno následující ilustrací:



Obrázek 1 Zjednodušený model Business Inteligence

3.3 Datový sklad

Za účelem rychlého získání aktuálních a přesných informací z mnoha nesourodých vstupních zdrojů byl zaveden tzv. Data Warehouse - datový sklad, jenž si nyní v rámci rešerše základních pojmů BI po teoretické stránce popíšeme.

Ve své podstatě se jedná o databázi, která na rozdíl od těch klasických relačních, ze kterých jsou ovšem data, obvykle z různých platforem či geografických lokalit, právě pro datový sklad v pravidelných intervalech sbírána předzpracována, nepodléhá takovým restrikcím, jakými jsou např. normálové formy. Nad daty z datového sklady se posléze vykonávají analýzy, jejichž výsledky shrnutí za co nejdelší časové období po zpracování koncovými uživateli, jakými jsou manažeři a analytici, mohou mít pro dané odvětví strategický význam pro současnost a budoucnost.

Poměrně známá a v mnoha publikacích užívána je formální definice autora Billa Inmona, přeložená např. v [3]:

Podnikově strukturovaný depozitář subjektově orientovaných, integrovaných, časově proměnlivých, historických dat použitých na získávání informací a podporu rozhodování, obsahuje atomická i sumární data.

Databáze, ze kterých datový sklad coby soubor technologií čerpá zdrojová data a přeměňuje na informace za účelem podpory dalšího rozhodování, označujeme jako produkční (procesní) či operační. Ve srovnání s nimi, resp. jejich typem databáze, tj. transakční databází, se datový sklad dále vyznačuje těmito rozdíly:

- Pomalejší odezva (obecně až o 3 řády horší, v nejlepším případě stejná),

- oproti DML určeno pouze pro čtení (manipulace s daty je pak ale jednodušší a odpadají metody pro optimalizaci, transakce apod.),
- původ dat je požadován až za neomezený časový úsek (časová variabilita bývá oproti operačním databázím hlavním ukazatelem)
- kategorizace dat podle subjektu, nikoliv aplikace,
- v nejlepším případě stejná, obvykle ale větší velikost datového fyzického prostoru díky velkému množství redundantních dat, indexů a předpočítaných hodnot.

S termínem datového skladu je úzce spjat také další - **datový trh**, jenž jsme zavedli výše při popisu BI. Jedná se o menší přesně definovanou část datového skladu pro menší odvětví a organizace, na jehož vstupu je jediný zdroj. Datový sklad se skládá z několika datových trhů, můžeme tedy o tzv. *data marts* prohlásit, že jsou podmnožinou právě datového skladu.

Zatímco způsob uložení dat v datovém trhu či procesních databázích bývá obvykle na bázi technologie OLTP, v datovém skladu to je pak díky velkému a neustále narůstajícímu objemu vstupních dat technologie OLAP, díky čemuž jsou data tvořící informace formou kladeného požadavku (dotazu) zpřístupněny v ideálním případě ihned, prakticky v mnohem kratším intervalu několika řádů, než-li by tomu bylo za použití prvně jmenované technologie OLTP.

3.4 Reportovací služby v prostředí MS SQL Server 2008

Pro tvorbu reportů v prostředí MS SQL Serveru 2008 je možno vybrat ze dvou hlavních a oficiálně podporovaných nástrojů:

- Business Intelligence Development Studio.
- Report Builder.

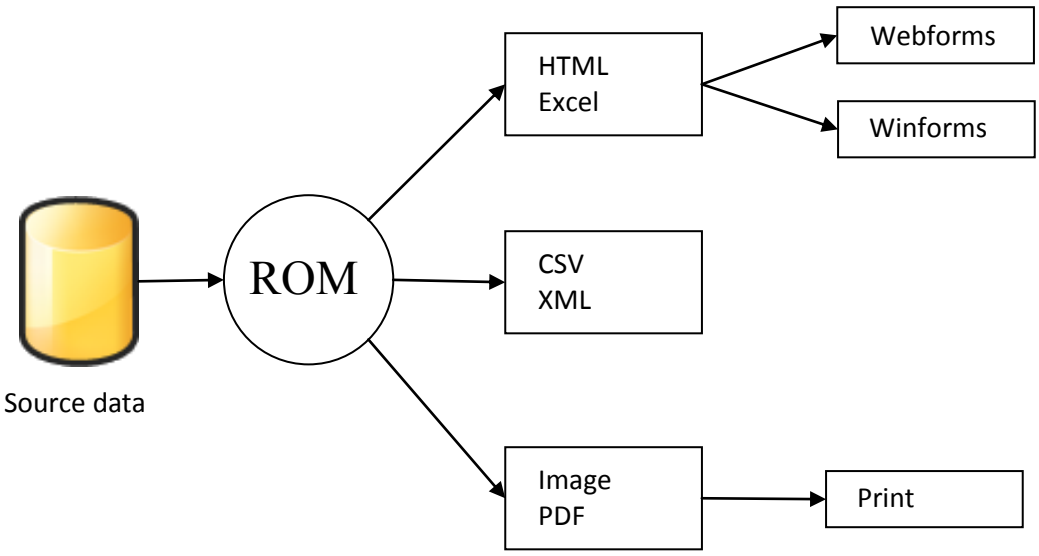
Prvně jmenovaný je svým původem, chováním, základními ovládacími prvky i vzhledem odvozen od nástroje MS Visual Studio, ovšem zaměřen pouze na BI. Ke svému chodu ovšem potřebuje verzi 2008 ve 32-bitovém prostředí (nikoliv 2010) a s ní spojené požadavky (např. .NET Framework).

Oproti tomu Report Builder, v současnosti (zima 2012) ve verzi pod číselným označením 3.0, MS Visual Studio nevyžaduje, koncept ovládání a vzhledu je převzat z kancelářského balíčku MS Office. Také zde je ovšem nutná přítomnost .NET Frameworku (ve verzi 3.5 s integrovaným SP 1 či vyšším).

Obě představené aplikace typu tzv. desktop patří do množiny nástrojů MS SQL Server 2008 R2 a na svém vstupu očekávají rozličný výčet formátů vstupních dat (primárně data SQL Serveru, předpřipravené datové kostky analytických služeb a nástrojů, data jiných SŘBD prostřednictvím ODBC a pod.).

Prostřednictvím interního průvodce a ovládacích prvků lze vytvořit interaktivní výstup na základě požadovaných, vyselektovaných a vyfiltrovaných dat na základě restrikcí, doplněný např. o grafické znázornění formou grafu, jehož tvorba a možnosti jsou opět podobně široké, jako při tvorbě samotných výstupních dat (tj. tabulka či seznam).

V rámci fáze vydání a publikování reportu lze pak výsledek uchovat v několika formátech. Konkrétně např. v rámci BIDS lze takový proces zachytit následujícím schématem,



Obrázek 2 Schéma procesu vytvoření reportu v BIDS

kde ROM je hlavní výkonná jednotka reportovací služby, jenž na základě předaných dat a informací vytvoří (Source data) jeden z ve schématu vyznačených druhů výstupů (HTML, XML, ...). Ten pak lze publikovat v rámci SQL Serveru a zpřístupnit sobě nebo ostatním uživatelům serveru v rámci webového rozhraní.

Součásti reportů je soubor RDL, což je podmnožina souborů rodiny XML, nesoucí vždy popisné informace (metadata) a také samotný obsah (formou vypsání přímo všech dat a informací, nebo formou SQL řetězce).

3.5 Reportovací služby v dalších prostředích

Mimo uvedené aplikace a jejich nástroje lze využít při tvorbě reportů a analýz také méně známé či preferované aplikace, resp. jejich součástí je také podpora BI:

- MS Office Excel 2007

Tvorba reportů tomto známém kancelářském produktu je ve formě kontingenčních tabulek nad různými typy datových připojení. Těmi mohou být OLAP datová kostka, tabulka z databáze MS SQL Serveru, Access, XML či jiným datovým připojením prostřednictvím OLEDB. Samozřejmostí je také podpora různého druhu formátování a přídavného grafu s prvky interakce uživatele (filtry, rozklikávací položky formou stromové struktury, ...) podobně, jako ve výše zmíněných nástrojích.

- MS Office Excel 2010

Novinkou v prostředí Excel 2010 pro podporu BI je oproti svému staršímu předchůdci integrace tzv. PowerPivot (přímo v aplikaci označeného jako Gemini) pro podporu mj. rychlého načtení a analýzy rozsáhlého objemu dat (v řádech desítek milionů záznamů), který do tohoto kancelářského nástroje přináší samostatné grafické prostředí. Pravděpodobně největší devizí se pak

stává analýza vztahů jednotlivých entitních typů, případně jejich dodatečná správa (vytvoření a odstranění), na jejichž základě lze lépe modelovat a procházet výslednou analýzu.

- Crystal Report

Tento nástroj, podporovaný podle [4] také v MS Visual Studio 2003, 2005 a 2008, je definován jako aplikace BI pro návrh a tvorbu reportů širokého spektra datových zdrojů. Těmi mohou být od základních (textové, XML, ...) až po celé databázové projekty (nativně podporované jsou typy Oracle, PostgreSQL, MS Access, MySQL a další), případně jiného zdroje pomocí tzv. datových mostů či providerů (ODBC, JDBC, ...).

- MS Office SharePoint Server 2010

V prostředí MS Excel vytvořené analýzy a reporty typu PowerPivot report (případně od jiných zdrojů) lze dále spravovat a využívat v prostředí MOSS 2010. To znamená správu takových dokumentů přes webové rozhraní informačního systému, tj. prostřednictvím internetového prohlížeče. Můžeme např. mimo prohlížení takového dokumentu nastavit obnovu dat, tedy zda vůbec a v jakých intervalech se v něm mají zdrojová data, nad kterými je prováděna analýza, aktualizovat, přičemž na výběr pak máme možnosti jako obnova dat podle časového intervalu (jednou, denně, týdně a měsíčně), času (v přesně definovanou hodinu a minutu), či zda se mají v případě různých datových zdrojů aktualizovat pouze námi vybrané.

- FastReport.NET

FastReport.NET je dalším z představitelů kategorie generátorů tiskových sestav určených pro operační systémy Windows s podporou pro vývojové prostředí Microsoft Visual Studio 2005 až 2010 a .NET Framework od verze 2.0, neboť i samotná aplikace reportovacího nástroje je vyvinuta v jazyce C#.

Program jako celek je kompletně přeložený mj. také do češtiny, což usnadňuje práci i méně zkušeným uživatelům a dopomáhá k intuitivnějšímu ovládání, ke kterému se ještě dostaneme. Pro nekomerční použití a lehké vyzkoušení jsme si obstarali volně šiřitelnou demoverzi, která plní svůj testovací účel. Plné verze programu je nutné v reálném světě dokoupit, ceny se liší v závislosti na délce licence, počtu klientských stanic či zvolené distribuce. Cena nejelementárnější z nich začíná přibližně na částce 2 000,- na jeden rok.

Program tedy není součástí žádného jiného kancelářského balíku známého z předchozích prací (SQL Server, Office, série Visual studio, operační systémy apod.) a můžeme jej tedy zařadit do kategorie softwaru tzv. **třetích stran**, tzn. takový software, který nám usnadňuje či rozšiřuje práci s jiným (původním, základním) softwarem, ale výrobci obou takových aplikací jsou různí a zároveň se nejedná o náš výtvor. Od takových dodatečných rozšíření se naopak vývojáři původního softwaru distancují bez záruky odpovědností v případě potíží a bývá často jen na dodavateli a distributorovi aplikace třetí strany, aby zajistil jeho správný chod vhodnou podporou po celou dobu platné licence (opravné balíčky, aktualizace, online podpora, nápovědní centrum, manuály apod.).

4 Rozšíření současného generátoru výstupních sestav

Jednotlivé rozšiřující prvky byly navrženy na základě studie výchozí diplomové práce, zejména jejího závěru, konzultaci s vedoucí této práce a vlastního uvážení.

4.1 Hlavní body rozšíření

Mnou navrhovaný a implementovaný generátor výstupních sestav vychází ze základů diplomové práce [1]. Na základě její studie jsem navrhnul následující hlavní linie, které budou charakterizovat rozšíření a odlišnosti od tohoto díla a mohou tedy k němu tvořit doplněk:

- Přidání nových typů uživatelských rolí a možnost přidání a správy nových prostřednictvím uživatelského rozhraní,
- další typy zdrojových, tj. vstupních dat (PostgreSQL, SQLite, Firebird),
- podpora poddotazů SQL na úrovni vytváření sestavy,
- na úrovni implementace zavedení návrhového vzoru MVC 3 a Repository v prostředí ASP.NET,
- využití nativní podpory zmíněných datových typů na výstupu, resp. ve výstupním programu jako alternativa k použitému ODBC v [1],
- Možnost práce se samotným SQL řetězcem.

Jednotlivé body výše uvedeného seznamu budou nyní blíže definovány a upřesněny v následujících podkapitolách.

4.2 Uživatelské role

V původním informačním systému se po procesu autentizace nacházejí dvě uživatelské role [1], kterými jsou Administrátor a Uživatel. Rozšířený systém pak bude obsahovat následující výčet typů uživatelských rolí:

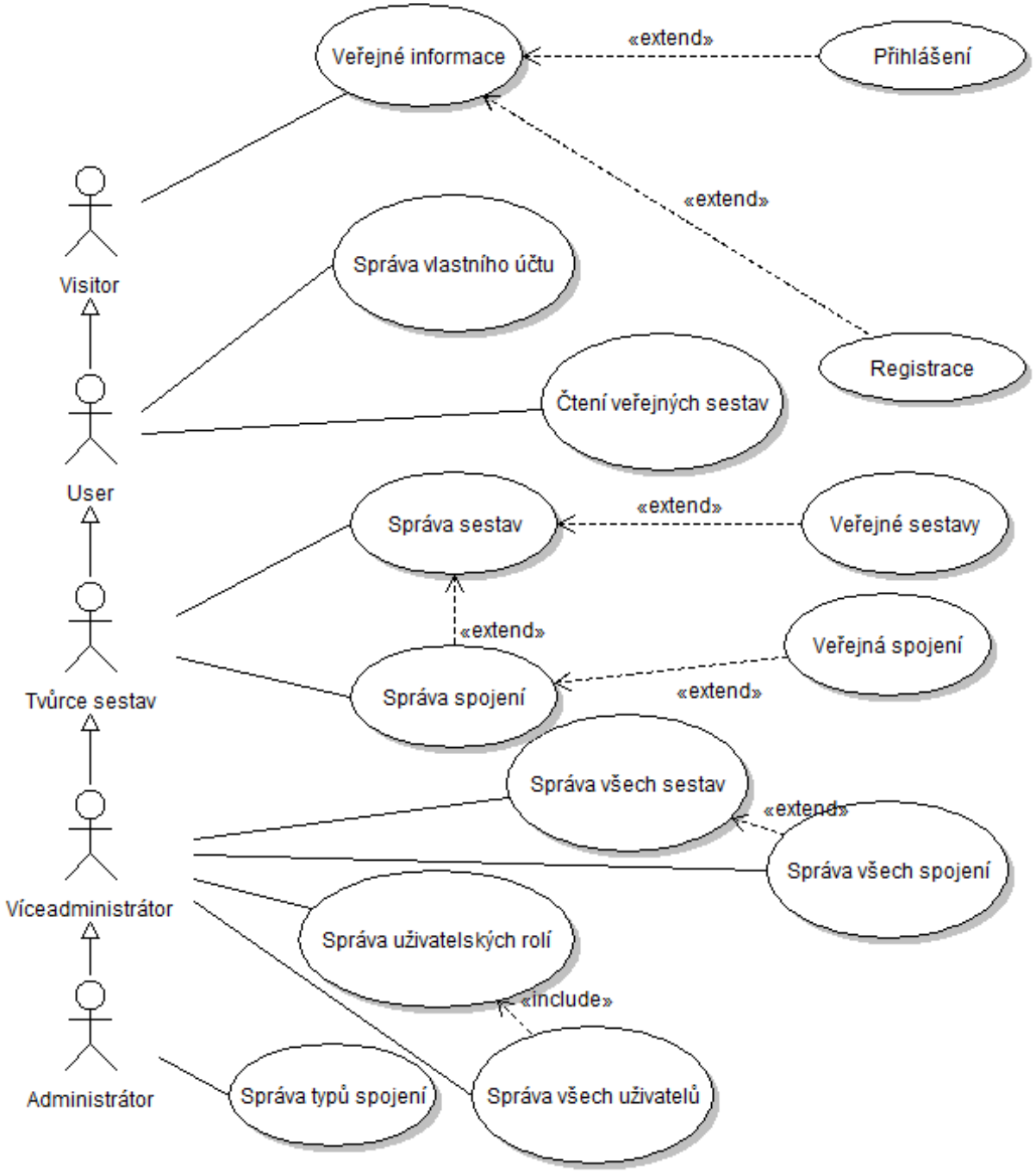
- Systémové,
- definované uživatelem.

Prvně jmenovaný typ bude obsahovat takové role uživatelů, které byly do systému integrovány od počáteční inicializace projektu, resp. databáze, a za standardních podmínek, tj. prostřednictvím uživatelského rozhraní, je nelze odstranit ani přidávat. Do této kategorie konkrétně patří následující uživatelské role

- Administrátor,
- Víceadministrátor,
- Tvůrce sestav,
- Uživatel.

Naopak druhý typ uživatelských rolí, tj. uživatelské role definované uživatelem v rozhraní informačního systému po jeho inicializaci, lze vkládat a také odstraňovat, názvy takových rolí pak volí uživatel s daným oprávněním dle svého uvážení. Předpokládá se, že takto navržený model rolí se využije zejména při správě oprávnění k jednotlivým sestavám, resp. k prohlížení jejich detailů.

Jednotlivé role jsou zachyceny také grafickou formou, konkrétně diagramem případů užití, tzv. Use Case Diagram (Obrázek 3).



Obrázek 3 Use Case diagram rozšířeného generátoru

4.3 Vstupní datové zdroje

Následuje charakteristika jednotlivých typů datových zdrojů.

4.3.1 SQLite

4.3.1.1 Úvod

Prvním z vybraných představitelů SŘBD, se kterým jsme na akademické půdě neměli prostor se seznámit a blíže si jej představíme v rámci této diplomové práce, je SQ Lite, často známý a dohledatelný také bez mezery pod výrazem SQLite. Za produktem stojí sdružení SQLite Consortium, ve které najdeme významné hráče na poli současného dění v oblasti vývoje informačních technologií, jakými jsou např. Oracle, Nokia, Adobe nebo Mozilla [5].

Obrázek 4 Logo SQLite

4.3.1.2 Základní princip

Hlavním základním kamenem systému je jednoduchost. V praxi se jedná o relativně malou knihovnu nevyužívající architekturu klient-server, ve nichž se databázový server instaluje a spouští jako samostatný proces, ale výsledný datový soubor s koncovkou mdf je připojen přímo s aplikačním programem, který nad tímto souborem vykonává operace.

4.3.1.3 Vývoj a platformy

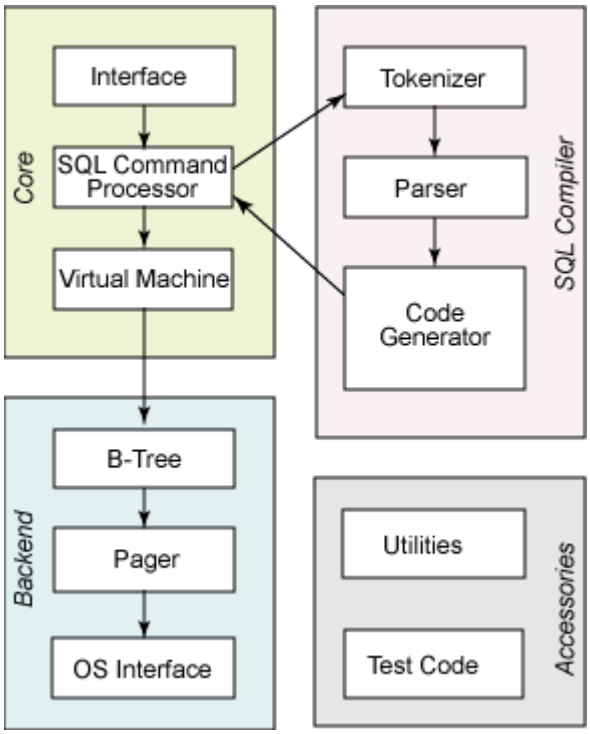
Ačkoliv by se mohlo zdát být zřejmé, že produkt SQLite neměl od svého počátku samotného vývoje, v polovině roku 2000 programovacím jazykem C, ambice stát se vysoce sofistikovaným a komplexním nástrojem pro správu datových souborů, bohatá historie vývoje jednotlivých verzí tohoto relačního systému a přidávaných funkcí z oficiálního pramenu udržuje tento projekt kompatibilní s dnešními platformami, kterými jsou Linux, Max OS X a Windows. Ve výsledku pak můžeme databázi SQLite pod licencí public domain využít v programovacích jazycích C (jádro systému), C++, Delphi, Java, Lua, PHP, Python, Perl, Ruby, Tcl. V Současné době (září 2011) je k dispozici verze s označením 3.7.8. Vůbec první spustitelná verze pro konečné zákazníky byla uvolněna jen pár týdnů po oficiálním oznámení počátku jeho vývoje, jehož počátek je připisován osobě jménem D. Richard Hipp (nar. 1961), tedy SW architektem a členem Tcl týmu spravující jeho jádro.

Jak bylo zmíněno výše, relační systém používá pro práci soubory typu MDM (akronym odvozen od Database Manager), představující jednoduché databázové rozhraní a jedná se o implementaci SQL jazyka právě nad těmito soubory, dodržující standard SQL92 pro zaručení práce s index, transakcemi a pohledy. Transakce mají přívlastek atomická (je spuštěna kompletní úspěšně, anebo vůbec), uzamčená (dvě a víc transakcí spuštěných nebo probíhajících současně se navzájem neovlivňují) , konzistentní a stálá (odolnost vůči výpadkům). Data do souborů ukládá vždy do stejně velkých bloků za využití jednoduchého klíče. Systém navzdory své původní beztypovosti postupem času a vývoje osahuje jen několik základních datových typům, nevýhodou je pak také v aktuální verzi absence např. datového typu pro datum a čas, který pak v případě potřeby musíme řešit programově a způsobuje tak větší režijní nároky na komunikaci úloh v programovacích jazycích s datovými soubory. Výjimku tvoří datový typ označen jako `integer primary key`, tedy jak název napovídá, vždy unikátní označení

v rámci daného typu objektu formou celého kladného čísla. Mezi další omezení patří absence cizích klíčů a závislostí.

4.3.1.4 Architektura

Jádro SQLite se skládá uvnitř z několika komponent, kterými jsou SQL kompilátor, jádro, rozhraní (backend) a doplňky (accessories, utilities).



Obrázek 5 Architektura SQLite

Po fyzické stránce je možno dále ukládat databáze, tedy datové soubory MDM až do velikosti 2 TB. Jedné databázi vždy odpovídá právě jeden soubor na disku strukturou B+ stromu, na který musíme mít povoleny práva zápisu.

Atributům můžeme přiřadit jeden z podporovaných datových typů, jakými jsou NULL, INTEGER, REAL, TEXT, a BLOB.

4.3.1.5 Distribuce a instalace

Již od počátku vývoje a vzniku prvních verzí byla snaha o podporu binárních předkompilovaných souborů pro různé platformy za účelem uspokojení a obsazení co možná nejširšího pole působnosti na trhu. Dnes (září 2011) si můžeme vybírat z několika typů distribucí, jakými jsou zdrojové soubory, dokumentace, předkompilované binární soubory pro Linux, Os X, Windows a .NET. Napříč každého typu si dále můžeme obvykle vybrat z několika nabízených možností (samotný spustitelný soubor, dll knihovna, nástroj pro analýzu a rekonstrukci provozu SQLite apod. Distribuce nejsou členěny na výběr mezi 32 a 64 bitovou platformu.

4.3.1.6 Porovnání velikostí vybraných distribučních verzí různých SŘBD

Velikost distribučního souboru SQLite pro náš ukázkový příklad ve zmíněné verzi 3 pro OS Windows zabírá několik desítek kB (252 kB). Systém MySQL dosahuje v závislosti na verzi a distribuce velikostních objemů 12.70 MB (verze 3.23), 25.4 MB (verze 4.0.26) či také nejvýše 37.68 MB (verze 5.1). Robustní verze systému Oracle se pohybují v řádech 100-ek MB.

4.3.1.7 Použití

V současnosti, tak, jako u řady ostatních SŘBD, existuje několik nástrojů pro správu samostatných databázových souborů tohoto typu. Vedle základního ovládání pomocí příkazové řádky operačního systému je k dispozici např. využití grafického uživatelského rozhraní, které nabízí i méně zručným uživatelům snadnější manipulaci nad daty např. formou internetového prohlížeče Mozilla Firefox, do kterého je dostupný volně šiřitelný instalovatelný doplněk SQLite Manager, v současnosti (září 2011) k dispozici ve verzi 0.7.6.

4.3.2 PostgreSQL

Druhým z celkových tří SŘBD, na které je v rámci této práce soustředěna pozornost v návaznosti na trojici databázových systémů v práci předchozího diplomanta, jenž je touto rozšiřována, je PostgreSQL.

4.3.2.1 Úvod

Název v následujících odstavcích popisovaného SŘBD bývá často označován zkráceně jako Postgree či Postgres, vycházející ze systému Ingres jako svého předchůdce z počátku 70. let v USA jak názvem, tak také funkčností i přes prvotní využití jazyka QUEL namísto pozdějšího SQL.

Obrázek 6 Logo PostgreSQL

V současnosti je chápán jako objektově relační databázový systém, hojně rozšířen a hostingovými společnostmi podporován nejenom u nás, ale také v zahraničí. Za svá léta existence a vývoje vyniká svou rychlostí, stabilitou, integrací i rozšiřitelností. V neposlední řadě pak také díky bohaté historie podporou jak ze stran tvůrců a producentů, tak širokou odbornou veřejností, jejichž blogy, diskusní fóra či rozšiřitelné prvky neustále přibývají, také v závislosti a v reakci na nově vydané distribuce.

Produkt je šířen pod licencí tzv. free a open source, podobně jako licence typu MIT či BSD, tedy v překladu jakýsi svobodný program s otevřeným zdrojovým kódem, volně šiřitelným, upravitelným a dále využivatelným jak po stránce implementační, tak po právní stránce, čímž se liší od svého komerčního předka Ingres.

Žádanou a obsaženou vlastností je také multiplatformost, kdy PostgreSQL můžeme provozovat nad systémy Linux, Unix a Windows.

4.3.2.2 Historie

Vývoj samotného Postgree započal v roce 1986 v kalifornské univerzitě University of California v Berkeley pod hlavou profesora Michalea Stonebrakera a jeho společnosti tvořenou zejména jeho studenty z doktorských programů. Společně věnovali na akademické půdě vývoji 8 let, kdy byly do systému postupně implementovány a definovány pravidla se souvisejícími entitními typy, procedury, rozšiřitelné datové typy, objektově-relační koncept a pod. Roku 1994 po vzdána celkem čtyř verzí program uvolnili a ukončili, načež vznikala jednak později snaha soukromých firem o proprietizaci produktu se začleněním do svého portfolia (Novel Netware, IBM, Informix, Great Bridge s Red Hat, ...) tak také hlavně zpočátku ihned po jeho ukončení na kalifornské univerzitě skupinami dobrovolníků, kteří jej dále vyvíjeli.

Roku 1995 byl nahrazen Stonebrakerovými doktory původní jazyk POSTQUEL (interpreter dotazovacího jazyka QUEL) za SQL a dílo pojmenováno jako Postgres95. Další nové označení získal tento software hned v následujících dvou letech 1996-97 nově jako PostgreSQL, číselně označována jako 6.0. Po funkční stránce se pracovalo především na stabilizaci původních univerzitních verzí, ale od verze 7.0 mluvíme v souvislosti s PostgreSQL také o standardu SQL92 s podporou indexů, spouštěčů, transakcí, cizích klíčů apod.

Po roce 2000 se k podpoře vývoje PostgreSQL přidávají společnosti jako EnterpriseDB, Command Prompt, Inc., či Sun Microsystems. K dnešnímu dni (podzim 2011) evidujeme verzi s označením 9.X.X. a je využívána velkými nadnárodními korporacemi (Cisco, The American Chemical Society, a další).

4.3.2.3 Specifické vlastnosti

PostgreeSQL mimo již některé výše zmíněné vlastnosti disponuje např. následujícími charakteristickými rysy:

- Unicode znaková sada.
- Pohledy (Views).
- Spouštěče (Trigery).
- Procedury.
- Zamykání.
- Rozhraní pro ODBC, JDBC, .Net, C, C++, PHP, Perl, TCL, ECPG, Python a Ruby.
- Replikace.
- Dědičnost mezi entitními typy.
- Datové typy .
- Současný víceuživatelský přístup typu MVCC (Multi-Version Concurrency Control), nahradil zamykání na úrovni tabulek.

4.3.2.4 Distribuce a instalace

V současnosti (podzim 2011) je PostgreSQL jako multiplatformní aplikace dodávána v rámci uceleného instalačního komplexního balíčku napříč hlavními a nejpoužívanějšími operačními systémy, jakými jsou Linux, Windows a Mac OS, a to jak pro verze s 32b jádrem, tak 64b [6] .

4.3.2.5 Přehledová tabulka aktuálních distribucí a jejich velikostí v kB

	Linux x86-32	Linux x86-64	Win x86-32	Win x86-64	Mac OS X
9.1.1	47379	48983	48150	48944	61403
9.0.5	49905	52007	48893	48833	70237
8.4.9	41295	47096	43219		61120
8.3.16	35062	36570	39414		54118

Tabulka 1 Souhrn velikostí posledních verzí PostgreSQL pro různé platformy

4.3.2.6 Vybrané vlastnosti vybraných vývojových verzí

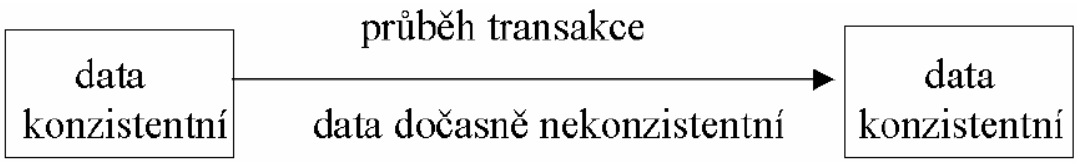
	7.4	**8.3**	**9.1**
UTF-8 (Win)	ne	ano	ano
Podpora MS VS C++	ne	ano	ano
Dočasný pohled (Views)	ne	ano	ano
2-fázové zamykání	ne	ano	ano
Změna datového typu atributů	ne	ano	ano
Informační SQL schéma	ano	ano	ano
Indexování výrazů	ano	ano	ano
Definice výchozích práv	ne	ne	ano
Víceřádkový příkaz pro vkládání	ne	ano	ano
SSL a IPv6	ano	ano	ano
Datový typ Enum a XML	ne	ano	ano
Omezení spojení na každého uživatele/db	ne	ano	ano
LDAP autentizace	ne	ano	ano
Fulltextové vyhledávání	ne	ano	ano
Hashování datového typu numeric	ne	ano	ano

Tabulka 2 Souhrn vybraných vlastností vybraných verzí PostgreSQL

SŘBD PostgreSQL plně podporuje výčet základních požadavků pro chod transakcí kladených na systémy tohoto typu a nese označení ACID, které v sobě nese následující body:

Atomičnost - v rámci spuštěné transakce se provedou buď všechny změny, nebo žádné (navrácení do původního stavu, resp. neprovedení požadovaných operací)

Konzistence - databázový systém je povinnen zaručit stálost měněných informací před i po provedení transakce, ovšem v průběhu vykonávání této dané základní jednotky zpracování mohou být data proměnlivá.



Obrázek 7: Proces zachování konzistence u transakcí

Zachování tohoto bodu v ACID paradigmatech je možno docílit různými metodami, jakými jsou např. metody zpožděné aktualizace, přímé aktualizace, logování do souborů pro obnovu stavu, či stínové stránkování, jejichž základní principy jsme si objasnili v průběhu studia v předmětech s databázovou tématikou a jejich implementaci jedné či jejich vzájemnou kombinaci již řeší každý SŘBD individuálně.

Izolace - chod transakce je jak logicky v návrhu imlementace, tak funkčně oddělen od ostatních transakcí (tj. jsou nezávislé na okolí). Požadujeme tedy, aby zatím nepotvrzené databázové změny dat byly neviditelné.

Trvanlivost - změny informací jsou v databázi po potvrzení vykonání transakce trvalé.

PostgreSQL je postavena na multigenerační architektuře MVCC (Multi Version Concurency control), ve které se namísto zamykání jednotlivých tabulek, řádků či stránek vytváří kopie takového modifikovaného objektu (nejčastěji celého řádku). Takových verzí může pak existovat libovolné množství v závislosti na vnitřním chodu transakce, ale každá transakce vidí vždy pouze jedinou verzi řádku, který v sobě navíc nese řadu stavových hodnot. Mezi nejdůležitější patří označení transakce, které danou verzi vytvořila a označení transakce, která kopii (verzi) řádku ruší. Každá verze je pak viditelná pouze právě tehdy, byla-li taková transakce potvrzená, anebo pokud se jedná o aktuální transakci (verze řádků jsou v PostgreSQL ukládány na konec tabulky). Kopie řádků se odstraňují v okamžik, kdy nejsou viditelné z žádné transakce příkazem VACUUM, který je specifický právě pro PostgreSQL.

Tento proces oproti klasickému zamykání přináší spolu s principem WAL (Write ahead logging) vyšší výkon, kdy jsou indexy i entitní typy jako datové soubory upravovány teprve po zapsání záznamu do logovacího transakčního souboru bez podmíněného diskového zápisu a jeho případné změně při neplatné transakci.

4.3.2.7 Datové limity

Maximální velikost každé jedné tabulky, tj. entitního typu, je v systému PostgreSQL omezena fyzickou velikostí na 32 TB. Ta pak dále mimo tento absolutní limit podléhá také dalším restrikcím, kterými jsou max. počet sloupců (názvů atributů) v závislosti na datovém typu na rozmezí 250 - 1600, každý jeden řádek pak na 400 GB a jedna položka pak 1 GB.

4.3.3 Firebird

4.3.3.1 Úvod

Posledním z trojice představovaných s naším generátorem výstupních sestav komunikujících systémů je SŘBD FirebirdSQL, zkráceně známý také jako Firebird. Tento poměrně mladý relační databázový systém v současnosti spravuje neziskové sdružení vývojářů zvané Firebird Foundation, z čehož plyne také právní povaha celého projektu jehož distribuce jsou aktuálně (podzim 2011) vydávány jako typ open source, v tomto případě konkrétněji nazvanou Initial Developer's Public License (IDPL).

Obrázek 8 Logo SŘBD Firebird

4.3.3.2 Historie

Samostatná historie produktu se sice datuje teprve od roku 2000, ale podobně jako u svých dříve popisovaných předchůdců také FirebirdSQL vzešel jako jedna z vývojových větví systému, jehož počátky sahají ještě o pár let dříve, v tomto případě se jedná o odnož systému z 80. let, komerčního systému InterBase vyvíjeného a spravujícího dodnes známou společností Borland, v tomto případě ovšem pod licencí uzavřených zdrojových kódů. Architektura InterBase systému dále vychází z dalšího databázového systému, tehdy nazvaného jako RDB, jenž s sebou přinesl multigenerační architekturu, kterou z velké části následně převzali systémy Postgre či Oracle.

4.3.3.3 Architektura

Podobně jako u Postgre je také Firebird postaven na základech architektury MVCC, s jejíž základním principem jsme se seznámili v předešlých odstavcích právě u PostgreSQL. Obdobně splňuje také zmiňovaná ACID pravidla pro práci s transakcemi či požadavky na multiplatformost, tzn. lze jej provozovat na systémech Linux, Windows, Mac OS, a to jak ve verzích 32b, tak 64b., přičemž velikost jednotlivých distribučních balíků je závislá na množství doplňkových modulů a typu instalace (SuperServer, Classic, Embeded), u tzv. "all in one" distribuce pro aktuální verzi 2.5 se pohybuje v rozmezí 5 až 25 MB. Podporovány jsou rovněž méně známe OS jakými jsou mj. Solaris či HP-UX. Datová velikost jedné databáze pak nesmí převýšit hodnotu 20 TB.

Je přítomna podpora v rámci připojení pro Delphi, C++, PHP, FireRuby, Firebird.NET a další. Ačkoliv je samotný SŘBD licencován jako open source, na poli doplňkových nástrojů třetích stran existuje široká škála programů a doplňků, z nichž některé z nich byly vyvinuty jako komerční produkt (např. IBExpert, Flame Robin apod.).

4.4 Podpora poddotazů SQL řetězce

Výsledkem našeho generátoru jsou sestavy, z nichž každá obsahuje jako svou hlavní část mj. samotný SQL řetězec, jenž je použit nad výše popsanými databázovými systémy. Ten, nazývaný v tomto smyslu také jako "vnitřní", oproti minulé práci [1], může obsahovat, v závislosti na požadavcích uživatele, který jej vytvořil, tzv. vnořený dotaz, tj. další direktiva typu SELECT ohraničená jednoduchými závorkami.

Ve své podstatě se obecně jedná o příkaz SELECT zanořený do dalšího, jenž vzniká především za účelem zjištění informace v závislosti na jiné informaci v databázi, která se musí nejprve zjistit. Teoreticky může takových zanoření v rámci jediného příkazu být libovolné množství a tak ani v našem generátoru není počet takových iterací omezen absolutním počtem, ale pouze v závislosti na tom, kolik restrikcí uživatel vytvoří v případě použití průvodce aplikace. V případě přednastavení vlastního SQL řetězce jako textový vstup pak není omezen ani vytvořenými restrikcemi.

Základní syntaxi a význam lze ilustrovat následujícím elementárním příkladem, kde chceme např. vypsat všechny data o produktech, které nebyly nikdy objednány (tj. nejsou obsaženy v žádné objednávce):

```
SELECT *
FROM Shop
WHERE Product.ProductID NOT IN
(
    SELECT Product.ItemID
          FROM Product, OrderItem
          WHERE Product.ProductID = OrderItem.ProductID
)
```

Obrázek 9 Elementární příklad poddotazu v SQL

Entitní typ OrderItem (ID, OrderID, ProductID, ...) představuje řádky všech objednávek v ukázkovém elektronickém obchodě Shop, typ objektu Product (ProductID, ...) pak reprezentuje samotný produkt (položku).

Podrobná rešerše a analýza příkazu SELECT pak můžeme najít v původní práci [1], ze které vycházíme.

4.5 ASP.NET MVC 3

S pomocí nástroje MS Visual Studio 2010 lze v prostředí ASP.NET vytvářet aplikace s návrhovým vzorem MVC, jenž je v takovém nástroji, společností MS označen v současnosti jako MVC 3, plně podporován a nabízí intuitivní ovládání pro správu jednotlivých komponent takového vzoru (kontrolery, modely, pohledy, případně oblast - tzv. area, apod.).

4.5.1 Controller

Soubory typu CS, po vzoru hlavní myšlenky zmíněného návrhového vzoru obsahuje výhradně třídy zdrojového kódu jazyka C#. Obvykle odpovídá jednomu kontroleru právě jedna třída a v právě jednom souboru v adresáři Controllers. Těch může být od verze MVC 2 v prostředí MS Visual Studio více v různých "lokalitách", v názvosloví nástroje MS se jedná o tzv. "Areas". Kontroler přijímá požadavky zvnějšku a přerozděluje práci , výsledky pak posílá do svého pohledu (Views) podle odpovídající metody, nebo tzv. přesměrovací tabulky definované v konfiguračním souboru projektu (webconfig.asax).

4.5.2 Model

Vrstva obsahující modely (abstraktní podoby entitních typů v připojené databázi) a repositáře, které tvoří prostředníka vždy mezi jedním kontrolerem a několika repositáři, kterým zasílá zprávy (jsou spolu v jednosměrné asociaci). Opět se jedná o zdrojové soubory v jazyce C#.

4.5.3 Views

Prezentuje data formou HTML, která mu přepošle kontroler, který je získá z repositářů. Změna pohledu nevyžaduje narozdíl od změn v předchozích dvou fragmentech překompilování celého projektu v případě změny v jednom z nich na bázi HTML. Samotný tvar a celé GUI pak záleží ta tazích popisného jazyka (X)HTML a kaskádových stylů CSS.

V prostředí MS Visual Studio 2010 je zabudován systém pro správu uživatelů, protože se ale jedná pouze o základní správu, budeme do IS zakomponovávat vlastní uživatelskou zónu tak, aby korespondovala s popsanými funkcemi. Skládá se ze 3 komponent: User Provided, Role Provider a Profile, nastavených v hlavním konfiguračním souboru web.xml v rootu každého projektu vytvořeného ve Visual Studio.

4.6 Přístup k datovým zdrojům

Pro potřeby tvorby reportů v prostředí MS SQL Serveru 2008 nad databázemi nejen typu i původně MSSQL, ale např. i MySQL bez převodu s mezistupněm v podobě např. exportu/importu kancelářským nástrojem MS Excel je potřeba konvertor těchto dvou typů SQL souborů dat.

Přístup k datům v konkrétním SŘBD aplikačních programů je rozdělen na dva způsoby:

- nativní podpora,
- prostřednictvím ovladače ODBC,
- prostřednictvím datových providerů.

Nativní podporou se myslí zabudovaná podpora konkrétního jazyka pro daný typ datového zdroje.

Prostřednictvím ovladačů a tzv. prostředníků (provider) lze pak k datům přistupovat i z původně nekompatibilních typů dat (např. jazyk C# prostředí .NET k datům obsažených v databázi typu Firebird), což je právě případ obou generátorů výstupních sestav, jak původního [1], tak navrhovaného a v dalších etapách implementovaného formou rozšíření. Narozdíl od prvně jmenovaného ovšem nebude využívat ovladače ODBC, ale datových prostředníků (providery), které v případě databázových zdrojů pro výše uvedené typy pracují s tzv. informačními schématy vždy dané již konkrétní databáze. Takové schéma pak obsahuje metainformace právě o dané databázi.

Ukázka vlastního konvertoru dat

Mimo samotný rozšířený generátor výstupních sestav byla provedena v rámci iterací také doplňková aplikace konvertoru dvou typů databází (MySQL a MSSQL), rovněž na platformě .NET a jazyka C#.

```
DVD:\Attachments\Convertor\Project\Application\Quick start
```

Umístění na DVD 1 Konvertor dat jako dopňková aplikace

Současná verze takového konvertoru (Umístění na DVD 1) očekává na svém vstupu datový soubor s exportovanou MySQL databází s definicí (schématy) jednotlivých typů objektů a případně také příkazy INSERT představující jejich řádky s daty. Formát a rozložení vstupu by mělo mít v ideálním případě např. takovouto podobu:

- Pro definici objektu,

```
CREATE TABLE IF NOT EXISTS `tableTitle` (
  `attributeTitle` attributeType[(typeSize)] [isNotNull]  [defaultValue] [isAutoIncrement],
  ...
  ...
  PRIMARY KEY [`primaryKeyTitle`] (primaryKeyArray),
   ...
  ...
  KEY [`keyTitle`] (keyArray),
  ...
  ...
) [otherSettings];
```

Obrázek 10 Syntaxe definice typu objektu potřebná pro konvertor

kde jednotlivé proměnné jsou ve významu:

tableTitle	povinný název typu objektu (např. Video).
attributeTitle	povinný údaj pro definici atributu nesoucí význam názvu atributu (např. Title)
attributeType	povinný údaj , jehož hodnota popisuje typ dat pro daný atribut (např. tinyint)
typeSize	proměnná omezující velikost (rozsah) dat daného datového typu (např. (256))
isNotNull	nepovinná proměnná pro povolení nulové hodnoty daného atributu (má hodnotu NOT NULL nebo žádnou)
defaultValue	nepovinný údaj určující počáteční hodnotu atributu s klíčovým slovem DEFAULT a hodnotu v uvozovkách pro každý objekt (např. DEFAULT '1', nebo DEFAULT NULL)
isAutoIncrement	nepovinný údaj, nejčastěji pro primární celočíselné atributy označující inkrementaci hodnoty implicitně od posledního záznamu, nabývá hodnoty AUTO_INCREMENT nebo žádnou)

primaryKeyTitle označuje nepovinné jméno primárního atributu

primaryKeyArray je pole názvů primárních existujících atributů v uvozovkách, oddělených čárkou, obvykle bývá velikost pole rovna právě jednomu prvku a platí pak vztah počet primárních atributů = počet řádků s definicí primárních atributů. Nejčastěji se jedná o celočíselné umělé primární atributy.

keyTitle, keyArray - analogicky jako primárních atributů, označuje položky (atributy) jenž se indexují

otherSettings - dodatečné informace o typu objektu v databázi (kódování, typ, počáteční hodnota automaticky inkrementálního atributu apod.)

- pro definici příkazu vkládání objektu (INSERT),

```
INSERT INTO ` tableTitle ` [(attributesTitles)] VALUES
(attributesValues),
...
(attributesValues );
```

Obrázek 11 Syntaxe vložení objektu pro konvertor

kde jednotlivé proměnné jsou ve významu:

tableTitle povinný název typu objektu (např. Video)

attributesTitles nepovinný seznam názvů atributů v uvozovkách oddělených čárkami, pořadí odpovídá pořadí při definici typu objektu (příkazu CREATE TABLE)

attributesValues je povinné pole hodnot pro jednotlivé atributy, které pak představuje jeden řádek (objekt), pořadí položek přiřazující atributu hodnotu odpovídá pořadí v poli attributesTitle a tranzitivně pak tedy zároveň pořadí atributů v definici typu objektu (CREATE TABLE). Hodnoty jsou odděleny čárkami, vyjma číselných a NULL hodnot v uvozovkách, každý řádek končí čárkou, poslední pak středníkem.

Takový formát výstupu nám vznikne např. při implicitním nastavení exportu databáze ve webové aplikace pro správu systému řízení báze dat (SŘBD) phpMyAdmin (v. 2.11.9.3), tj. s následujícími vlastnostmi:

- přidání příkazu pro kontrolu existence (IF NOT EXISTS),
- přidání automatické inkrementace (AUTO_INCREMENT),
- použití zpětných uvozovek u jmen tabulek a sloupců,
- úplné inserty,
- rozšířené inserty,
- použití šestnáctkového zobrazení pro BLOB,
- maximální velikost vytvořeného dotazu na hodnotu 50000.

Mezi definicemi typu objektu a vkládání do nich dat může být libovolná sekvence např. komentářů nebo prázdných řádků, které budou ignorovány.

```
DVD:\Attachments\Convertor\Input\
```

Umístění na DVD 2 Ukázka vstupních souborů pro aplikaci konvertoru typů dat

V příloze jsou příklady touto cestou originální - vstupní (Umístění na DVD 2) i transformované databáze, resp. skriptů, nad kterými byla aplikace pro konverzi testována (Umístění na DVD 3). První z nich je databáze videoportálů z mé bakalářské práce, druhý pak elektronický obchod dodaný vedoucím této diplomové práce za účelem otestování a dodání výsledku. Předpona "in" u každého souboru označuje originální (nezměněná) data, předpona "out" pak výsledný soubor konvertoru.

```
DVD:\Attachments\Convertor\Output\
```

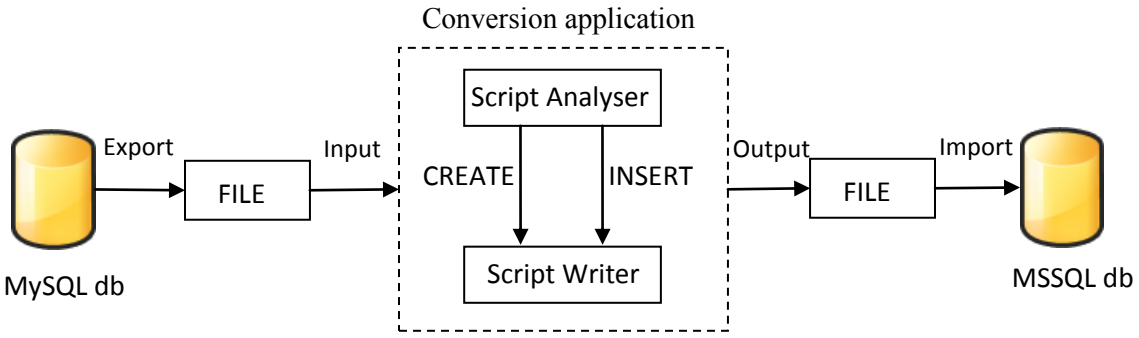
Umístění na DVD 3 Výsledné soubory konvertu typů dat

Samotný algoritmus konvertoru lze ve stručnosti charakterizovat následovně:

1) hledá všechny definice příkazu CREATE TABLE,

2) hledá všechny definice příkazu [INSERT].

Pro každou fázi projde znovu celý datový soubor, proto je aplikace nezávislá na jejich případné záměně pořadí. Celý proces včetně chování aplikace lze zachytit schématem:



Obrázek 12 Schéma konvertoru

Jedná se v principu o tzv. parsování vstupního (zdrojového) textu, kdy se snažíme převést data na jinou požadovanou (cílovou) formu pro další potřeby zpracování, v našem případě to je tvorba reportů nad daty v databázi. Přístup aplikace vzhledem ke zpracovávanému vstupu můžeme dělit na:

1. bajtový (výstup je prakticky nezávislý na uspořádání a formátování vstupu, čtení po bajtech, univerzální),
2. řádkový (vstup je zpracováván v sekvencích řádků, požaduje přísné pravidla pro formátování obsahu zdrojových dat).

V obou případech se ovšem předpokládá validní vstup s SQL příkazy.

Protože výsledkem této práce není přímo konvertor pro převod mezi dvěma typy databází, využil jsme řádkového zpracování dat s kombinacemi bajtového přístupu. Ve své aktuální verzi detekuje datové typy tinyint, mediumint, int, bigint, smallint, text, date, datetime, text, char, varchar, dále primární atributy, indexy, počáteční hodnotu atributu, atribut s přírůstkovým klíčem (auto increment, resp. identity), a zmíněné vkládání dat.

Aplikace očekává vstupní soubor formou prvního parametru včetně přípony, obdobně výstupní soubor třetím parametrem, který bude ve znakové sadě UTF 8, druhý parametr představuje název výsledné databáze v MSSQL (např., viz.Obrázek 13).

```
C:\Users\Alexei\Desktop\Diplomka\Iteration 4\Source\Application>"SQL Conversion.exe" vs
tup5.sql test1 output5.sql
```

Obrázek 13 : Ukázka spuštění aplikace s parametry

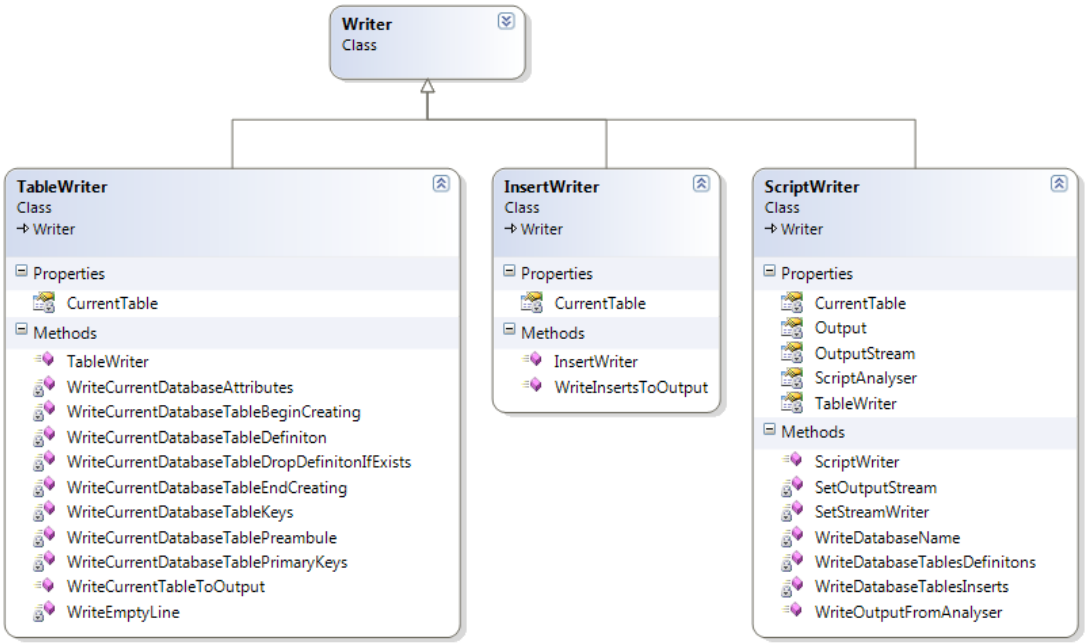
Ačkoliv se popisovaná aplikace ovládá výhradně prostřednictvím příkazové řádky, navrženo bylo rovněž GUI (Příloha C).

V elektronické příloze jsou pak rovněž kromě zdrojových kódů přiloženy doprovodné diagramy (Umístění na DVD 4),

```
DVD:\Attachments\Convertor\Diagrams\
```

Umístění na DVD 4 Diagramy ke konvertoru

z nichž jeden z nich je zobrazen jako ukázka také v této tištěné práci (Obrázek 14).



Obrázek 14 Jeden z diagramů konvertoru dat

4.7 Správa samotného SQL řetězce

Základním prvkem každé sestavy v prostředí navrhovaného generátoru bude SQL řetězec vygenerovaný na základě

- několika kroků v průvodci aplikace (od základních vlastností, výběrů tabulek, atributů, až po závorkování) bez nezbytně nutné větší znalosti z oblasti databází,
- nastavením řetězce SQL formou jednoho vstupního pole bez průvodce (např. z jiné aplikační úlohy nad danou databází).

Je zřejmé, že první způsob obnáší řadu mezikroků s důrazem na uživatelskou přívětivost (posloupnost kroků, vhodně zvolené GUI s ovládacími prvky apod.), zatímco druhý způsob je sice po stránce implementační nejsnadnější, na druhé straně se zde předpokládá již zkonstruovaný řetězec (např. v jiném programu).

5 Analýza rozšířeného generátoru výstupních sestav

Navrhovaný nový generátor výstupních sestav je blíže specifikován následujícím výčtem analýz, při kterém bylo, podobně jako v rámci dalších etap vývoje informačního systému, využito poznatků ze školních předmětů a k nim příslušících materiálů, např. v [7], [8], či v [9].

- Datová analýza
- Funkční analýza

5.1 Datová analýza

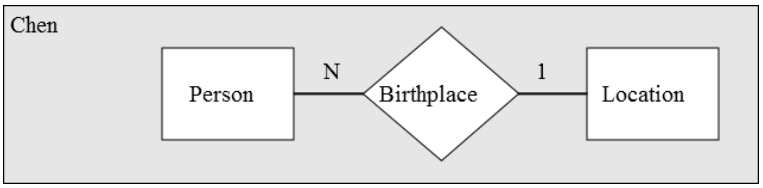
Datová analýza je v procesu tvorby informačního systému chápána jako jeden ze tří modelů, jehož výsledkem je konceptuální model systému. Takový model obsahuje následující položky:

- lineární zápis typů entit,
- lineární zápis typů vztahů,
- grafický diagram datových typů a typů vztahů (ERD),
- datový slovník,
- další integritní omezení.

5.1.1 Úvod k ER diagramům

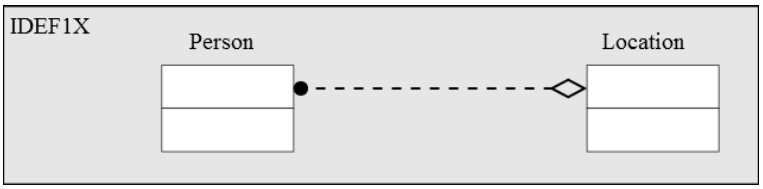
Jedním z nástrojů, jak grafickou podobou znázornit relační databázi, je ER diagram, zkráceně ERD, do které se zanášejí entitní typy s naznačením vzájemných závislostí (propojování, kardinalita, povinnost členství, apod.). Pro jejich tvorbu je možno využít několika základních notací, které lze případně vzájemně kombinovat, nemělo by se tak ovšem dít na úkor přehlednosti a jednoznačnosti.Mezi nejznámější patří [10]:

- Chenova notace



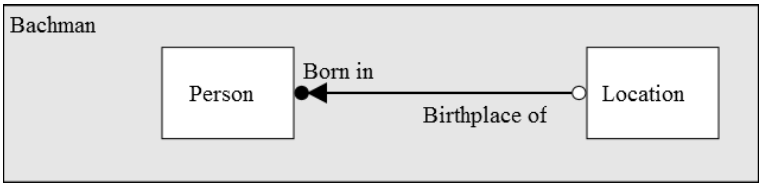
Obrázek 15 Chenova notace ERD

- IDEF1X



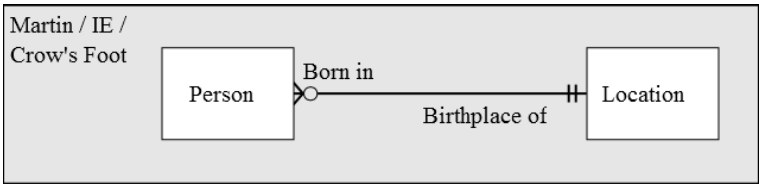
Obrázek 16 ERD notace IDEF1X

- Bachmanova notace



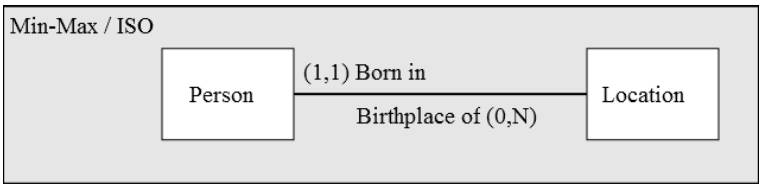
Obrázek 17 Banchmanova notace ERD

- Martinova notace



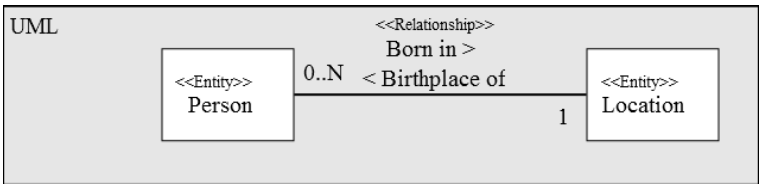
Obrázek 18 Martinova notace ERD

- notace Jean-Raymonda Abriala



Obrázek 19 ERD notace Jean-Raymonda Abriala

- Standard UML



Obrázek 20 ERD notace Standard UML

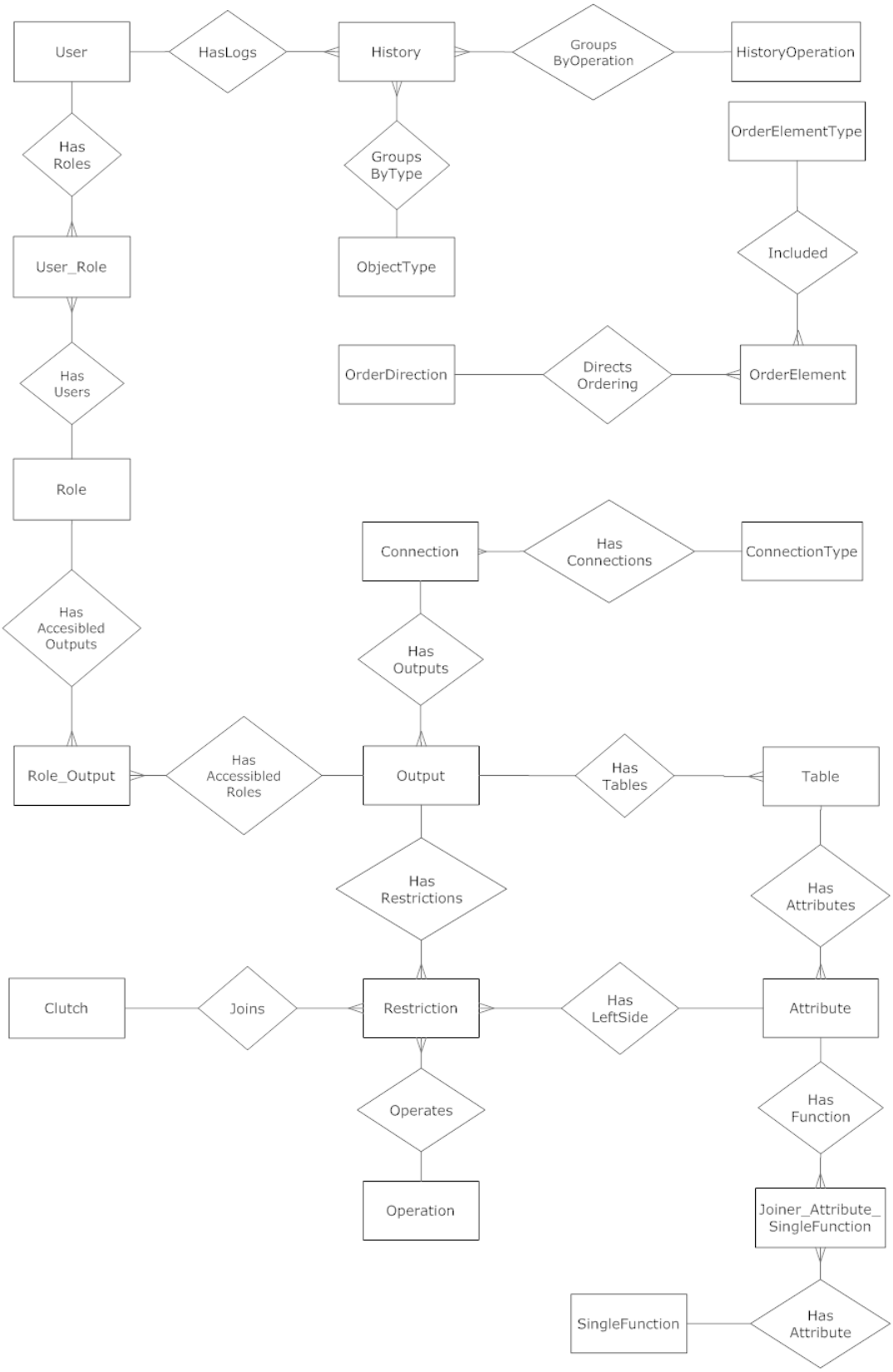
- EXPRESS

5.1.2 ER Diagram rozšířeného generátoru sestav

Kompletní datová analýza pro analyzovaný rozšířený generátor je přiložena na elektronickém médiu jako příloha (Umístění na DVD 5), zde uvádím, vedle linerárních zápisů typů entit a typů vazeb (níže), jen ERD (Obrázek 21).

```
DVD:\Attachments\Data Analysis\DA.pdf
```

Umístění na DVD 5 Datová analýza



Obrázek 21 ER Diagram rozšířeného generátoru

Podle výše zmíněné klasifikace typů notací ER diagramů je zřejmé, že tento konkrétní diagram spadá do prvně jmenované kategorie, tj. Chenova notace, známá z roku 1974.

5.1.3 Lineární zápisy

Grafickou podobu konceptuálního modelu analyzovaného systému doplňuje textový popis formou seznamu entitních typů a typů vztahů.

Takový popis se pak skládá z následujících druhů seznamů:

- lineární zápis typů entit
- lineární zápis typů vztahů

Z dokumentu datové analýzy zde uvádím formou dalšího výtahu konkrétní lineární zápisy jak pro entitní typy, tak také pro typy vztahů.

5.1.3.1 Lineární zápis typů vztahů rozšířeného generátoru

Legenda: **název entitního typu**, primární klíč, cizí klíč, název atributu

Attribute (AttributeID, TableID, Title, DataType, Priority, Alias, IsVisible, IsGroup, GroupPriority, IsOrder, OrderPriority)
Clutch (ClutchID, Title, Code, Priority)
Connection (ConnectionID, Title, Port, Login, Password, ConnectionTypeID, IsPublic, Database, Server, FilePath)
ConnectionType (ConnectionTypeID, Title, Priority)
History (ID, UserID, ObjectTypeID, ObjectID, HistoryOperationID, DataTime, Note)
HistoryOperation (HistoryOperationID, Title, Code, Priority)
Joiner_Attribute_SingleFunction (ID, AttributeID, FunctionID, Alias, Priority, IsVisible)
ObjectType (ObjectTypeID, Title, Code, Priority, Description)
Operation (OperationID, Title, Code, Priority)
OrderDirection (OrderDirectionID, Title, Code, Priority)
OrderElement (ID, OrderElementTypeID, ObjectID, Priority, OrderDirectionID)
OrderElementType (OrderElementTypeID, Title, Code, Priority)
Output (OutputID, Title, Description, IsPublic, ConnectionID, StartIndex, Count, CustomSQL)
Restriction (RestrictionID, OutputID, OperationID, LeftOperandID, RightOperandType, RightOperandID, RightOperandCustomValue, ClutchID, Priority, LeftBracketsCount, RightBracketsCount)
Role (RoleID, Title, Priority, IsSystem)
Role_Output (ID, RoleID, OutputID, Datetime)
SingleFunction (FunctionID, Title, Description, Priority, Code)
Table (TableID, Title, OutputID)
User (UserID, FirstName, SurName, Login, Password, Eaddress, IsArchive)
User_Role (ID, UserID, RoleID, Datetime)

5.1.3.2 lineární zápis typů vztahů rozšířeného generátoru

HasLogs (User, History) 1:M

HasRoles (User, User_Role) 1:M
HasUsers (Role, User_Role) 1:M
HasAccessibledOutputs (Role, Role_Output) 1:M
HasAccessibledRoles (Output, Role_Output) 1:M
HasOutputs (Connection, Output) 1:M
HasConnections (ConnectionType, Connection) 1:M
HasTables (Output, Table) 1:M
HasRestrictions (Output, Restriction) 1:M
Joins (Clutch, Restriction) 1:M
HasLeftSide (Attribute, Restriction) 1:M
HasAttributes (Table, Attribute) 1:M
Operates (Operation, Restriction) 1:M
HasFunction (Attribute, Joiner_Attribute_SingleFunction) 1:M
HasAttribute (SingleFunction, Joiner_Attribute_SingleFunction) 1:M
DirectsOrdering (OrderDirection, OrderElement) 1:M
Included (OrderElementType, OrderElement) 1:M
GroupsByType (ObjectType, History) 1:M
GroupsByOperation (HistoryOperation, History) 1:M

5.2 Funkční analýza

Funkční analýza je po analýze datové dalším, tedy druhým typem analýzy ve vývoji informačního systému. Jak je zřejmé z jejího názvu, popisuje funkce, které daný systém obsahuje a rozvádí je v daném kontextu do hloubky, tj. charakterizuje s jakými daty daná funkce pracuje a kterým aktérům přísluší. Samotná funkční analýza tvoří se dále dělí na dva typy:

- vnější pohled
- vnitřní pohled

5.2.1 Vnější pohled

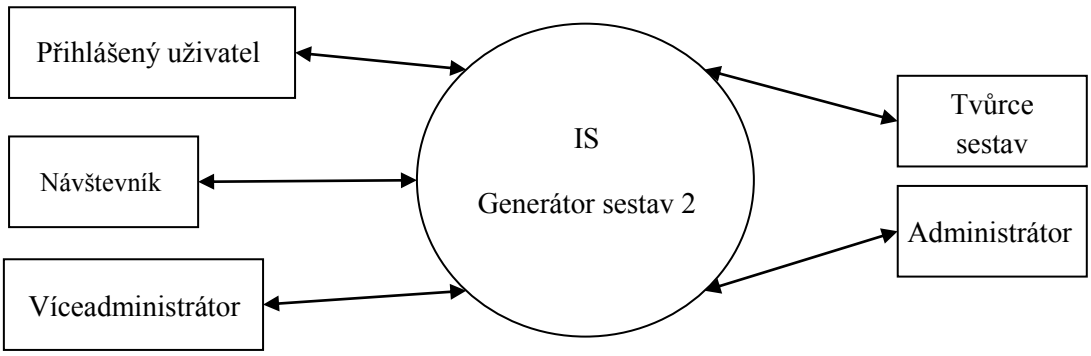
Hrubé grafické ztvárnění struktury funkčnosti informačního systému spadá do vnějšího pohledu funkční analýzy. Pohled se dále rozděluje do dalších úrovní, jejichž hloubka detailů postupně roste. Funkce na poslední úrovni jsou obvykle zdrojem pro vnitřní pohled. V rámci jednotlivých úrovní jsou funkce modelovány pomocí tzv. diagramů datových toků, neboli DFD diagramů (Data Flow Dragram), který tak popisuje transformaci dat využitím prvků, kterými jsou funkce, paměti, datové toky a aktéři.

5.2.1.1 Kontextový diagram

Nejvyšší úrovní vnějšího pohledu je kontextový diagram. Ten znázorňuje celý systém jako jedinou funkci, ale je základem pro další úrovně vnějšího pohledu. Může do značné míry korespondovat s tzv. aktivitním diagramem.

```
DVD:\Attachments\Function Analysis\FA.pdf
```

Umístění na DVD 6 Funkční analýza



Obrázek 22 Kontextový diagram

Z konkrétního kontextového diagramu (Obrázek 22) je patrné, že se systémem budou pracovat následující aktéři

- Návštěvník.
  Aktérem s názvem Návštěvník rozumíme kteréhokoliv neautentizovaného uživatele či vnější systém. Prvně jmenovaná možnost se bude vykytovat např. zejména v případě nasazení systému do některé z veřejných sítí (www, podniková síť, apod.). Druhou možností jsou pak např. automatizované systémy a skripty (např. vyhledávače, antivirové systémy apod.).

- Přihlášený uživatel.
  Poté, co uživatel projde v systému procesem autentizace, tzn. úspěšně vloží své jméno a heslo, které systém vyhodnotí jako správné, se z něj z pohledu funkční analýzy stává aktér s názvem Přihlášený uživatel.

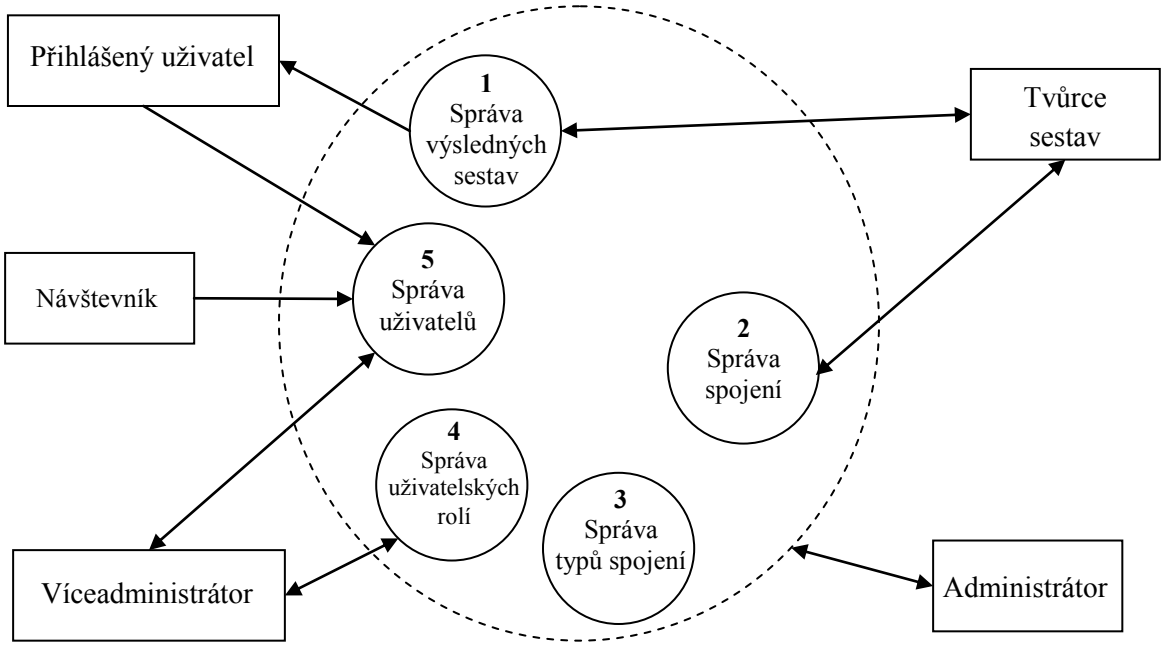
- Víceadministrátor.
  Aktér spravující všechny uživatelské účty a role se nazývá Víceadministrátor.

- Administrátor.
  Nejvyšší aktér v systému, tzn. aktér se všemi právy všech předcházejících aktérů.

5.2.1.2 Úroveň 0

Na kontextový diagram nadále navazující je tzv. DFD diagram 0. úrovně (Obrázek 23). Blíže rozkládá celý systém na další subsystémy, v tomto případě na pět hlavních celků. Aktéři jsou shodní s předchozím diagramem. V dalších rozkladech těchto částí již není označen aktér s názvem Administrátor, neboť se předpokládá, že využívá funkce a možnosti všech ostatních aktérů.

V kompletní funkční analýze je dále uveden rozklad až do 2. úrovně, které lze mj. zhlédnout v elektronické příloze.



Obrázek 23 DFD diagram 0. úrovně

5.2.2 Vnitřní pohled

Nejnižší úrovně vnějšího pohledu funkční analýzy jsou vstupem pro tzv. minispecifikace. Jedná se o algoritmy, které textovou formou doprovází diagramy a blíže tak popisují danou funkci.

Nyní uvedu výtah dvou vybraných minispecifikací, které, mimo ostatní, lze rovněž zhlédnout v přiloženém dokumentu na elektronickém médiu (Umístění na DVD 6).

5.2.2.1 Výběr tabulek

1. Načti a zobraz seznam dostupných sestav (tabulka Output).
2. Uživatel - víceadministrátor vybere objekt (sestavu), ke které chce přiřadit tabulky.
3. Ulož ID vybraného objektu (sestavy) do proměnné M_OutputID.
4. Přečti záznam typu spojení (tabulka ConnectionType, Connection, Output), kde Connection.ConnectionID = Output.ConnectionID, OutputID = M_OutputID, Connection.ConnectionTypeID = ConnectionType.ConnectionTypeID.
5. Vytvoř seznam všech databázových tabulek Tables ze sytémového katalogu (datový provider SQL Provider závislosti na typu spojení z kroku 4).
6. Vytvoř seznam všech dostupných databázových tabulek OutputTables (z tabulky Table), kde OutputID = M_OutputID.
7. Zobraz formulář "Výběr tabulek".
8. Pro každý záznam v seznamu Tables
   8.1. Vytiskni záznam z Tables na výstup.
   8.2. Pokud existuje záznam (podle Title) v seznamu OutputTables
   Pak

Vytiskni možnost přidání záznamu (typu Table) k sestavě (typ Output) formou seznamu TablesTitles s obsahem odkazující na název objektu (Title) ze seznamu Tables.

Jinak

Vytiskni možnost odebrání záznamu (typu Table) ze sestavy (typ Output)

9. Uživatel - tvůrce sestavy vybere zatržením objekty (tabulky), které chce k sestavě přiřadit.
10. Pro každý záznam v seznamu Tables dělej,

Pokud je takový záznam v seznamu TablesTitles (podle Title)

Pak

Vlož nový objekt do tabulky Table s proměnnou názvu (Title) aktuálně iterovaného záznamu seznamu Tables to tabulky Table (Title, M_OutputID).

Jinak pokud není v TableTitles a je v seznamu OutputTables

Odstraň z tabulky Table objekt, kde OutputID = M_OutputID a Title = TablesTitles.Title .

11. Zobraz zprávu o úspěšném provedení databázové operace.

5.2.2.2 Výstupní atributy

1. Načti a zobraz seznam dostupných sestav (tabulka Output).
2. Uživatel - tvůrce sestavy vybere objekt (sestavu), ke které chce určit atributy tisknutelné na výstup sestavy.
3. Ulož ID vybraného objektu (sestavy) do proměnné M_OutputID.
4. Vytvoř seznam atributů Attributes přiřazených k sestavě (tabulky Attribute, Table), kde Attribute.TableID = Table.TableID a zároveň Table.OutputID = M_OutputID .
5. Zobraz formulář "Výběr atributů na výstup".
6. Pro každý záznam v seznamu Attributes

6.1. Vytiskni záznam z Attributes do formulářové tabulky.

6.2. Pokud IsVisible = 0

Pak

Vytiskni možnost přidání záznamu (typu Attribute) na výstup, tj. změnu na IsVisible = 1

Jinak

Vytiskni možnost odebrání záznamu (typu Attribute) ze výstupu, tj. změna na IsVisible = 0

7. Uživatel - tvůrce sestavy vybere zatržením objekty (atributy), které chce přiřadit na výstup výsledné sestavy a odešle požadavek na server.
8. Ulož odeslaný upravený seznam do proměnné M_Attributes
9. Pro každý záznam s názvem M_Attribute v M_Attributes,

Přečti z tabulky Attribute záznam RealAttribute, kde AttributeID = M_Attribute.AttributeID.

Pokud RealAttribute.IsVisible != M_Attribute.IsVisible

Pak

Změň objekt v tabulce Attribute, kde AttributeID = M_Attribute.AttributeID, hodnotu atributu IsVisible na hodnotu M_Attribute.IsVisible.

10. Zobraz zprávu o úspěšném provedení databázové operace.

6 Návrh implementace rozšířeného generátoru výstupních sestav

Po ukončení specifikací, resp. ukončení většiny hlavních specifikací či určené sekvence specifikací v daném časovém úseku v případě iterativního vývoje, následuje etapa návrhu implementace, na jejíž vstupu je analýza navrhovaného systému, na výstupu pak bližší upřesnění pro samotnou implementaci, tj. vstup pro implementaci IS.

- Popis implementačního prostředí,
- zpřesněné datové struktury,
- zpřesněné a doplněné algoritmy funkcí systému,
- návrh dekompozice řešení na malé moduly,
- počáteční inicializace dat.

6.1 Implementační prostředí

Hardware, hardware serveru

Procesorová jednotka (CPU)
Intel Core 2 Duo T8400 3.00 Ghz

Operační pamět
8 GB RAM

Disk
Vzhledem k malému očekávanému množství multimediálních souborů (fotky, videa, zvuky) v řádech desítek MB.

Zobrazovací jednotka
17" s rozlišením 1280 x 1024 pixelů (nebo lepším, tj. vyšším).

Software

Operační systém
MS Windows Vista 64bit CZ .

Vývojový programovací jazyk
C# (C sharp), ASP.

Rozhraní
.NET 3.5, MS Visual Studio 2010 SP 1, MVC extension, JQuery framework .

Vývojové prostředí
MS VS 2010.

Architektura cílového projektu
klient - server.

Kompatibilita prohlížečů (testované verze)
Opera,11+ MS Internet explorer 9+, Firefox 4+.

6.2 Indexová analýza

Vytváření indexů nad daty v databázi zkracuje dobu potřebnou k jejich nalezení či provedení jiné operace. Přínosem je především pro data obsáhlého objemu v závislosti na výkonu stanice, kde je databázový systém nasazen. Protože tyto indexy, resp. ukazatele, požadují pro své uložení fyzický paměťový prostor, nedoporučuje se tedy používat indexy nad všemi daty v databázi.

V případě mého nového generátoru výstupních sestav je v datovém slovníku u každého atributu všech entitních typů uveden mj. také příznak s názvem "Index" nabývající hodnoty ano, nebo ne, jenž znamená přidání indexu k atributu, resp. nepřidání.

Takto označené indexy jsem u jednotlivých atributů volil především v těch případech, kdy se jedná o:

- primární klíč,
- cizí klíč (spojení s jiným entitním typem),
- atribut, podle kterého se provádí setříděný seznam,
- atribut, podle kterého se často vyhledává a nabývá malého množství výčtů hodnot (např. příznaky s logickými hodnotami).

6.3 Transakční analýza

Ve víceuživatelských informačních systémech je nutné v etapě návrhu implementace najít, popsat a ošetřit případy, kdy se ke stejným datům snaží dostat více než jeden uživatel. Protože výstupem mé diplomové práce je IS prozatím určený spíše pro akademický provoz za účelem testování na lokální stanici či lokální sítě s možností dalšího rozšíření s velmi malou pravděpodobností přihlášení více než jednoho uživatele a protože použitý SŘBD využívá transakce, nebude transakční analýza realizována.

Přesto pro formálnost uvedu příklad umístění zámků pro hypotetický případ, který by popíral důvody uvedené v předchozím odstavci, tj. víceuživatelská aplikace a nepodpora transakcí v SŘBD, a to na příkladě ukázkové minispecifikace uvedené výše v rámci ukázky z funkční analýzy doplněnou o příkazy zamčení a uzamčení. Je patrné, že se v ní vyskytují jak zámky sdílené, označené zkratkou LS, tak zámky pro tzv. výlučný přístup, označený jako LX. Označení pro odemčení je pak pro oba typy zámků stejné, a sice UN.

Příklad transakční analýzy doplněné o zámky pro minispecifikaci výběru výstupních atributů

LS (Output)

1. Načti a zobraz seznam dostupných sestav (tabulka Output).

UN (Output)

2. Uživatel - tvůrce sestavy vybere objekt (sestavu), ke které chce určit atributy tisknutelné na výstup sestavy.
3. Ulož ID vybraného objektu (sestavy) do proměnné M_OutputID.

LS (záznamy z tabulky Table, kde OutputID = M_OutputID)

LS (záznamy z tabulky Attribute, kde TableID = Table.TableID)

4. Vytvoř seznam atributů Attributes přiřazených k sestavě (tabulky Attribute, Table), kde Attribute.TableID = Table.TableID a zároveň Table.OutputID = M_OutputID .

UN (záznamy z tabulky Table, kde OutputID = M_OutputID)

UN (záznamy z tabulky Attribute, kde TableID = Table.TableID)

5. Zobraz formulář "Výběr atributů na výstup".
6. Pro každý záznam v seznamu Attributes
   6.1. Vytiskni záznam z Attributes do formulářové tabulky.
   6.2. Pokud IsVisible = 0
        Pak
            Vytiskni možnost přidání záznamu (typu Attribute) na výstup, tj. změnu na IsVisible = 1
        Jinak
            Vytiskni možnost odebrání záznamu (typu Attribute) ze výstupu, tj. změna na IsVisible = 0
7. Uživatel - tvůrce sestavy vybere zatržením objekty (atributy), které chce přiřadit na výstup výsledné sestavy a odešle požadavek na server.
8. Ulož odeslaný upravený seznam do proměnné M_Attributes
9. Pro každý záznam s názvem M_Attribute v M_Attributes,

LS (záznam z tabulky Attribute, kde AttributeID = M_Attribute.AttributeID)

Přečti z tabulky Attribute záznam jako RealAttribute, kde AttributeID = M_Attribute.AttributeID.

LS (záznam z tabulky Attribute, kde AttributeID = M_Attribute.AttributeID)

Pokud RealAttribute.IsVisible != M_Attribute.IsVisible
Pak

LX (záznam z tabulky Attribute, kde AttributeID = M_Attribute.AttributeID)

Změň objekt v tabulce Attribute, kde AttributeID = M_Attribute.AttributeID, hodnotu atributu IsVisible na hodnotu M_Attribute.IsVisible.

UN (záznam z tabulky Attribute, kde AttributeID = M_Attribute.AttributeID)

10. Zobraz zprávu o úspěšném provedení databázové operace.

6.4 Instalace IS a inicializace databáze

Instalace nového generátoru (dále pouze generátoru) výstupních sestav na stanici serveru probíhá přesunem všech zdrojových souborů přiložených v elektronické příloze. Součástí je také skript generující databázi v případě, pokud nebude možné využít zdrojový databázový soubor, jenž je integrován k projektu. Skript obsahuje

- vytvoření všech tabulek a jejich definice vč. atributů
- naplnění počátečními (inicializačními daty)
- ukázkové příklady pro seznámení se s prací průvodce sestav

Základem grafického uživatelského rozhraní je původní šablona ASP.NET MVC 3, doplněná o mnou vytvořené, nebo volně šiřitelné grafické prvky a styly.

6.5 Změny oproti původní analýze

Při implementaci je doporučeno zavést k takovým entitním typům, u kterých se předpokládá vyhledávání podle jejich autora, přidat atribut odkazující na takového uživatele (např. UserID). Zmiňovanými typy jsou např. `Output` a `Connection`. Důvodem je příliš náročný proces takového dotazu podle návrhu v analýze, tj. přes entitní typ `History`.

Uživatelské role oproti diagramu případů užití budou přejmenovány do anglického jazyka.

```php
<?php
  $connection = new PDO("pgsql:dbname=Eshop;host=localhost", "postgres", "admin");
  $sql = '...';
  $select = $connection->prepare($sql);
  $select->execute();
  $colcount = $select->columnCount();

  // meta information (header)
  for($i = 0; $i < $colcount; $i++):
    $meta = $select->getColumnMeta($i);
    // do any, for example print:
    print $meta['name'];
  endfor;

  // content
  // for each row
  foreach ($connection->query($sql) as $row):
    $j = 0;
    // for each column in row
    foreach ($row as $key=>$value):
    if($j%2 == 1):
      // do any, for example print:
      print $value;
    endif;
    $j++;
    endforeach;
  endforeach;
?>
```

Obrázek 24 Zkrácená ukázka vygenerovaného kódu pro testovací sestavu

7 Implementace

Podobně jako ve výchozí diplomové práci [1], také v této se v etapě implementace využilo platformy ASP.NET a OOP jazyka C#. Využito bylo zejména principů z [11] a [12]. Kompletní hlavní projekt, tj. samotný generátor, je uveden v elektronické příloze včetně databáze se základními (testovacími) daty (Umístění na DVD 7).

```
DVD:\Generator\Project\
```

Umístění na DVD 7 Rozšířený generátor výstupních sestav

Implementaci doprovází programátorská příručka formou HTML/TEX dokumentace, jenž byla vygenerována na základě komentářů přímo ve zdrojových souborech (Umístění na DVD 8).

```
DVD:\Attachments\Programmer's Guide\
```

Umístění na DVD 8 Programátorská příručka

Projektovou strukturu lze rozdělit na následující hlavní části (moduly):

- uživatelská zóna (správa uživatelů),
- spojení (k serveru a databázi, k datovému souboru apod.),
- typ spojení (typ SŘBD na vstupu, resp. výstupu),
- průvodce sestavy (správa sestav),
- ostatní (doprovodné stránky, globální nastavení, ...).

Každý z uvedených modulů pak ve struktuře obsahuje své vlastní prvky návrhového vzoru MVC, tj. své vlastní kontroléry a pohledy.

Výsledný generátor byl, mimo průběžných dílčích testů, vyzkoušen na komplexním příkladě pokrývající většinu z kroků průvodce sestavy. Vstupem byla ukázková databáze zjednodušeného elektronického obchodu nad vybraným typem databáze, v tomto případě PostgreSQL a jenž je umístěna spolu s ostatními testovacími vstupními daty v elektronické příloze (Umístění na DVD 9).

```
DVD:\Generator\Input\eshop-postgreSQL.sql
```

Umístění na DVD 9 Zdrojová databáze testovacího příkladu vybrané databáze (Postgre)

Hlavním cílem testovaného případu byl výpis všech zákazníků, kteří si ještě neobjednali žádný produkt, tj., nejsou uvedeni v žádné objednávce jako zákazník. V tomto požadavku je tedy pokryt následující výčet realizovaných vlastností průvodce sestavy:

- základní vlastnosti sestavy (název, počet zobrazovaných objektů, popis, počáteční pozice, ...),
- výběr databázového spojení,
- výběr tabulek prostřednictvím provideru pro daný druh databáze,
- výběr atributů prostřednictvím provideru pro daný druh databáze,
- výběr výstupních atributů,

- zástupné jména atributů,
- pořadí elementů na výstupu (atributů i případných funkcí),
- vytváření sady podmínek (restrikcí typu WHERE),
- řetězení podmínek (pořadí a spojky),
- vytvoření poddotazu v podmínce,
- volitelně oprávnění k sestavě (resp. k detailu sestavy),
- vygenerování a stažení výsledného skriptu.

V projektu je tato testovací sestava uložena pod názvem Test vnořeného dotazu I. Ta využívá také jinou sestavu (Seznam objednávek III) jako vnořený dotaz zkonstruovaný prostřednictvím průvodce generátoru a jeho GUI.

```
DVD:\Generator\Output\Output 79.php
```

Umístění na DVD 10 PHP skript vytvořený generátorem Test vnořeného dotaz I

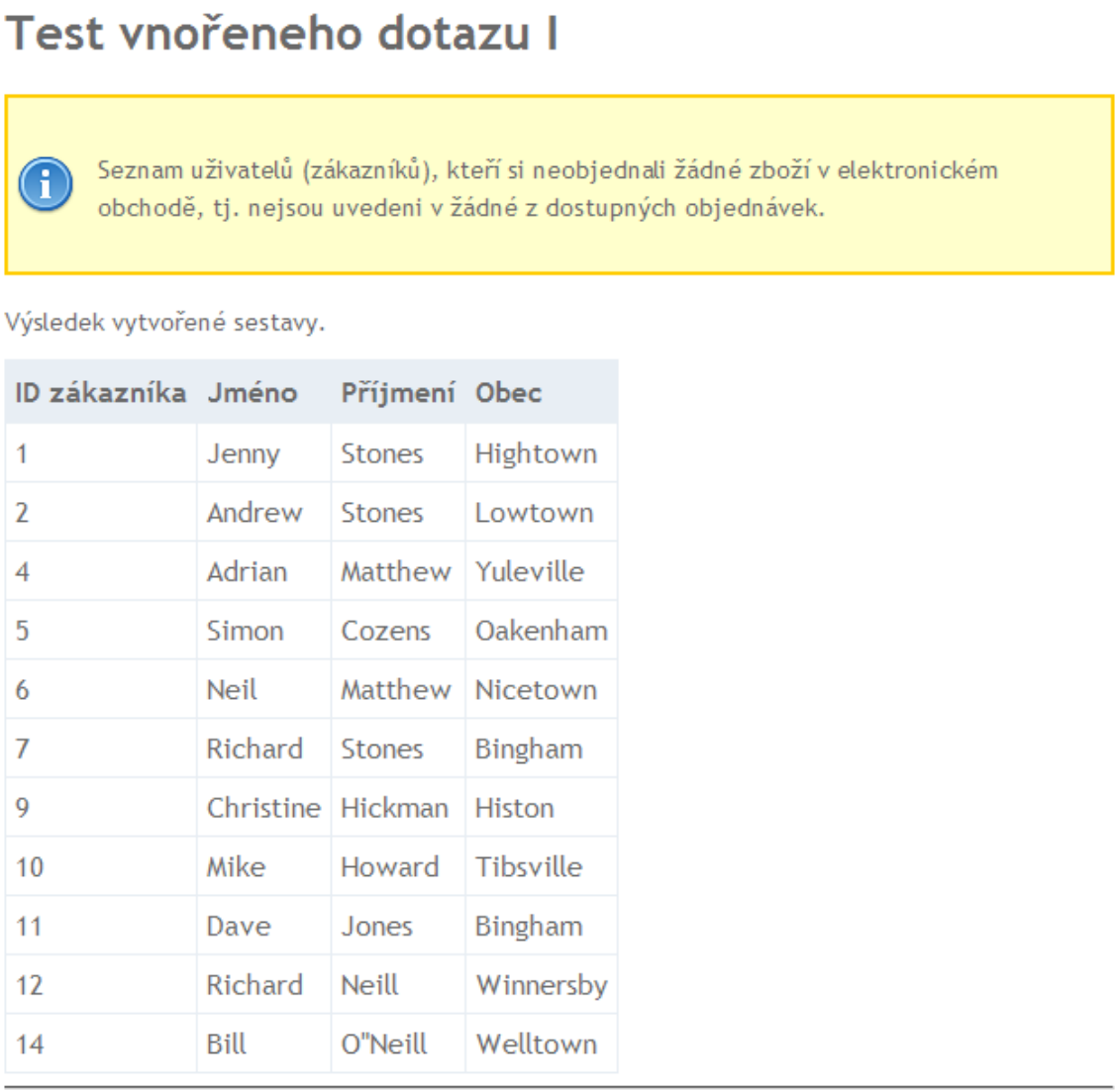
Generátor pak vyhotovil následující SQL řetězec (Obrázek 25), který je pak základním stavebním kamenem při tvorbě skriptu v jazyce PHP. Tento řetězec je pak vstupem pro rozhraní PDO, jejíž zkrácená ukázka bez HTML značek a validací je vidět rovněž v ilustraci (Obrázek 24). Tento automatický generovaný soubor je jako ukázka přiložen zvlášť v elektronické příloze (Umístění na DVD 10). Jako rychlou ukázku je část výsledku přiložena také zde formou obrázkové ilustrace (Obrázek 26).

```
SELECT
    customer.customer_id AS "ID zákazníka",
    customer.fname AS "Jméno",
    customer.lname AS "Příjmení",
    customer.town AS "Obec"
 FROM customer
 WHERE customer.customer_id NOT IN
     (SELECT  customer.customer_id
      FROM customer,orderinfo
      WHERE customer.customer_id =  orderinfo.customer_id
      OFFSET 0 LIMIT 100)
 ORDER BY customer.customer_id
 OFFSET 0 LIMIT 30
```

Obrázek 25 Ukázka SQL řetězce vytvořeného generátorem

Výsledné skripty v jazyce PHP, generované v prostředí generátoru implementovaného v ASP.NET, využívá vzhledem k možnému přístupu až ke třem typům databázových zdrojů tzv. datového prostředníka (provider) označeného v technologii PHP jako PDO. Předpokladem je pak korektní

nastavení serveru, na kterém bude vygenerovaný PHP skript spouštěn, pro zmíněné typy databází (PostgreSQL, SQLite, Firebird).



Test vnořeneho dotazu I

Seznam uživatelů (zákazníků), kteří si neobjednali žádné zboží v elektronickém obchodě, tj. nejsou uvedeni v žádné z dostupných objednávek.

Výsledek vytvořené sestavy.

ID zákazníka	Jméno	Příjmení	Obec
1	Jenny	Stones	Hightown
2	Andrew	Stones	Lowtown
4	Adrian	Matthew	Yuleville
5	Simon	Cozens	Oakenham
6	Neil	Matthew	Nicetown
7	Richard	Stones	Bingham
9	Christine	Hickman	Histon
10	Mike	Howard	Tibsville
11	Dave	Jones	Bingham
12	Richard	Neill	Winnersby
14	Bill	O"Neill	Welltown

Obrázek 26 Výsledek ukázkového příkladu pro vnořený dotaz

Pro srovnání uvádím také příklad reportů vytvořených v dalších nástrojích prostředích, jakými jsou např. MS SQL Server 2008, FastReport apod. (Umístění na DVD 11), a to formou přímo zdrojových souborů (tj samotný projekt), exportovaného souboru (formátu docx pro MS Office Word. Opět se jedná o příklad prováděný nad databází z mé bakalářské práce, konkrétně nad IS zabývající se videoreportážemi z oblasti zdravotnictví.

```
DVD:\Attachments\Other Tools Reports
```

Umístění na DVD 11 Reporty vytvořené v dalších nástrojích

7.1 Testování

Testování je nedílnou součástí vývoje SW aplikací v oblasti IT a nejinak tomu je také v případě IS. Můj generátor provázela v průběhu jeho vývoje řada testů, zejména z oblasti manuálních, z nichž vybrané jsou přiloženy v elektronické příloze (Umístění na DVD 12). Jako ukázka je pak přiložen jeden z nich také v této práci viz níže, konkrétně se jedná o počáteční proces vytvoření sestavy.

```
DVD:\Generator\Testing\Manual Testing.pdf
```

Umístění na DVD 12 Vybrané testovací scénáře rozšířeného generátoru

Legenda k vybraným testovacím scénářům:

Ikona	Popis
	Úspěch. Testovaný případ proběhl v pořádku, skutečná událost odpovídala očekávánému jevu a proces nebyl ničím přerušen.
	Varování. Testovaný případ proběhl, ale s výhradami, které ovšem nezabránily dalšímu pokračování v testovacím scénáři.
	Neúspěch. Testovaný případ neproběhl a chod aplikace byl přerušen. Testovací scénář v takovém bodě selhal a výsledný scénář nelze vyhodnotit jako úspěšný.

7.1.1 Ukázka scénáře z přiložených scénářů - Založení nové sestavy

Název scénáře	Založení nové sestavy
Autor	Martin Dočkal
Datum vytvoření	20.4.2012
Podmínky	• Spuštěn server. • Internetový prohlížeč Opera 11.62 na klientské stanici. • Spuštěná aplikace generátoru sestav. • Přihlášený uživatel v roli Administrátor,Víceadministrátor, nebo Tvůrce sestav. • Existence alespoň jednoho spojení s vnější databází (typ objektu `Connection`). • Zobrazena stránka pro první krok průvodce sestav (Základní vlastnosti).
Vstupy	Informace o sestavě.
Očekávané výstupy	Vložení nového záznamu do databáze . Zobrazení zprávy o úspěšném vložení.
Skutečné výstupy	Vložení nového záznamu do databáze . Zobrazení zprávy o úspěšném vložení.

Testovací případy

ID	Popis	Výsledek
1	Zobrazení formuláře se poli pro název, popis, viditelnosti, výběru spojení, počátečního indexu, maximálního počtu objektů, odkazu na CSS a tlačítkem Uložit pro odeslání formuláře na server.	
2	Vyplnění formuláře testovacími údaji.	
3	Odeslání formuláře stiskem tlačítka Uložit.	
4	Vložen nový záznam s daty z formuláře do tabulky Output.	
5	Zobrazení hlášení o úspěšném vložení do databáze.	

Výsledek testu: Test proběhl v pořádku.

8 Uživatelská příručka

Dokument, který svému čtenáři napoví a případně na jednoduchém příkladě ukáže, jak danou aplikaci zprovoznit a udržovat, se nazývá uživatelská příručka, se kterou se můžeme setkat nejenom v oblasti SW v informačních technologiích. Bývá často doprovázena řadou ilustrací zachycených z běžného užívání popisovaného předmětu nebo události.

8.1 Úvod

Vzhledem k obsáhlosti uživatelské příručky k rozšířenému generátoru výstupních sestav je umístěna do elektronické přílohy formou PDF dokumentu (Umístění na DVD 13), z něhož si zde uvedeme pouze jako ukázku výtah vybraných kapitol.

```
DVD:\Attachments\User's Guide\Manual.pdf
```

Umístění na DVD 13 Uživatelská příručka

Dokument pak tvoří následující strukturu

- Obsah
- Požadavky
- Instalace
- Struktura GUI generátoru
- Ovládání aplikace generátoru
- Chybová hlášení
- Reference

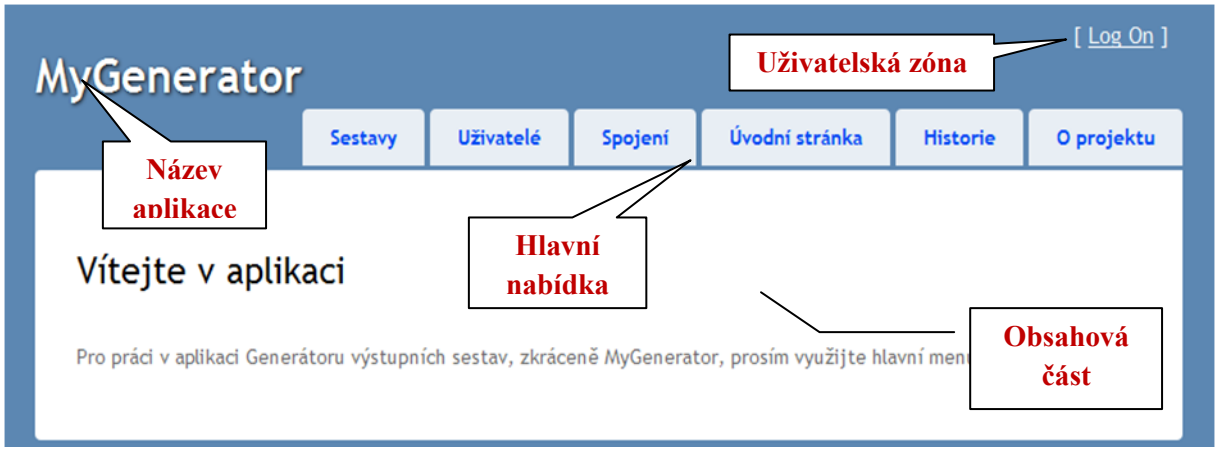
Je zřejmé, že bodu zabývajícím se ovládáním samotné aplikace generátoru je věnována větší pozornost, než ostatním.

8.2 Vybrané kapitoly z uživatelské příručky

Následuje zmiňovaný výtah několika segmentů z uživatelské příručky.

8.2.1 Struktura GUI generátoru

Po spuštění webové aplikace rozšíření generátoru se za splněných podmínek (požadavky, autentizace, ...) zobrazí následující stránka, kterou je možné vidět na obrázku (Obrázek 27).



Obrázek 27 Základní struktura aplikace

Rozmístění a vzhled jednotlivých částí se může lišit v závislosti na typu prohlížeče, operačního systému klientské stanici, velikosti zobrazovací jednotky, uživatelské role (ne)přihlášeného uživatele a pod.

Grafické uživatelské rozhraní se skládá z následujících základních prvků:

- Hlavní menu
- Hlavní obsahová část (*dále dělitelná v závislosti na typu vybrané stránky a operace*)
- Uživatelská sekce (*určeno především pro přihlášení nebo odhlášení aktuálního uživatele*)

8.2.2 Správa sestav

8.2.2.1 Seznam sestav

Seznam dostupných sestav (Obrázek 25[1]) si ze serveru vyžádáme např. mj. kliknutím na položku `Sestavy` v hlavní nabídce aplikace. Jednotlivé objekty jsou řazeny podle data vytvoření vzestupně. U každé sestavy v takovémto výpisu je vidět jednoznačné identifikační číslo sestavy (`ID`), její název (`Název`), Název a typ spojení (`Spojení`), autor sestavy (`Autor`), datum a čas vytvoření a další možnosti formou hypertextových odkazů.

8.2.2.2 Detail sestavy

Podrobné informace o sestavě (Obrázek 26[1]) nalezneme na samostatné stránce po kliknutím např. na hypertextový odkaz `Detail` v seznamu sestav. Zobrazí se nám tak informace o samotné sestavě, databázovém spojení, použitých elementech v průvodci sestavy, historie sestavy a také např. další možnosti práce se sestavou formou hypertextových odkazů, které nás přesměrují na příslušný obsah v závislosti na našem výběru (stažení skriptu, nastavení vlastního SQL řetězce, spuštění průvodce sestavou apod.).

[1] Reference na obrázek je v daném kontextu myšleno odkazem do uživatelské příručky jako originálního dokumentu v elektronické příloze, nikoliv v rámci tohoto textu diplomové práce.

8.2.2.3 Průvodce vytvoření nebo editace sestavy

Průvodcem sestavy rozumíme sadu 12 hlavních kroků, které nám umožňují formou grafického uživatelského rozhraní sestavit požadovaný SQL řetězec, který posléze můžeme využít buď samostatně, či stažením skriptu, který aplikace na jeho základě vygeneruje.

Spustíme jej např. kliknutím na hypertextový odkaz `Průvodce` v seznamu sestav vybraného objektu, či v detailu dané sestavy. Formuláře jsou shodné jak při vytváření nové sestavy, tak při editaci již existující, ve druhém případě jsou pak ale zobrazena data, která byla nastavena při poslední práci s průvodcem dané sestavy.

8.2.2.4 Základní vlastnosti

Po spuštění průvodce jsme přesměrování na první z 12 kroků. Server po nás formou formuláře vyžaduje vyplnění či editaci základních vlastností sestavy a především pak databázového spojení (Obrázek 28[1])

8.2.2.5 Restrikce

V sedmém kroku průvodce vytvářením sestavy má uživatel, danou sestavu tvořící, výhodu, pokud má alespoň základní informace v oblasti databází. Konkrétně se jedná o vytváření jednotlivých podmínek (Obrázek 36[1]), kde každá se skládá z levé strany, typu operace a volitelně pak pravé strany, jenž je závislá na typu pravého operandu, který uživatel nastaví v dalším dodatečném formuláři kliknutím na hypertextový odkaz `Zvolit typ` (Obrázek 37[1]). Na základě jeho výběru nám průvodce pro pravou stranu nabídne jednu z následujících možností:

- textové pole (pro případ **číselného** typu nebo **řetězce**)
- seznam sestav (v případě zvolení výstupu jiné **sestavy**)
- seznam atributů v rámci sestavy (v případě zvolení **atributu**)

Každou restrikci vytváříme v levé části aplikace tohoto kroku, v pravé pak můžeme vidět již existující dříve vytvořené restrikce s možností jejich odstranění (Obrázek 38[1]), přičemž po nás v takovém případě server požaduje potvrzení v dodatečném formuláři (Obrázek 39[1]).

9 Porovnání navrženého generátoru s ostatními

V této kapitole je porovnán mnou navržený a implementovaný generátor výstupních sestav jak s jeho původní předlohou, kterou rozšiřuje [1], tak také s dalšími službami.

9.1 Srovnání s původním generátorem

Můj generátor se snaží na původní (vzorovou) diplomovou práci [1] navázat, resp. rozšířit a klade důraz na odlišující se prvky. Jedná se např. o zmíněné jiné datové zdroje, implementace podle aktuálního trendu v technologii ASP.NET MVC, rozdílný přístup k datovým zdrojům jak na vstupu, tak výstupu, a změny ve funkčnosti generátoru i jeho vzhledu a snaha o vyšší stupeň uživatelské přívětivosti, o které se mj. zmiňuje autor původního generátoru v závěru své práce [1]. Jednotlivé položky jsou blíže specifikovány v závěru této práce, tj. v následující kapitole.

9.2 Srovnání s ostatními generátory

Ostatní, zejména komerční generátory či nástroje pro podporu BI (reportovací a analytické služby, reportovací servery a repositáře) v sobě skrývají pro uživatele nepřeberné množství možností, jak výslednou sestavu nebo report jednak vytvořit, ale také jak dále uchovávat a spravovat. Mezi výhody takových produktů oproti mému generátoru patří zejména

- doprovodné grafy a jejich,
- grafická škálovatelnost výstupu,
- možnosti formátování výstupních dat (do tabulek, seznamů, apod.),
- větší pokrytí typů dotazů pro výstupní sestavu (tj. pro řetězec SQL),
- množství integrovaných funkcí (pro práci s daty, řetězci, ...).

Mimo tyto zmíněné vlastnosti, ve kterých oba generátory ztrácejí oproti jiným zabudovaných v komerčních nástrojích, dále zaostávají v segmentech vyplývajících z kapacitních možností vývojových týmů daného generátoru (tj. implementace multijazyčnosti, rozsáhlé dokumentace a podpora, testování).

Formát rozšířeným generátorem vytvářených skriptů výstupních sestav zůstal shodný, jako v jeho předloze, a to je v PHP. Další reportovací nástroje umožňují ukládat výsledek do rozličných formátů (např. Excel, PDF, HTML, ...), nicméně takový bod nebyl ani v rozšiřujících bodech na počátku budování rozšíření nastaven a nebyl tedy ani realizován. Můžeme ovšem předpokládat, že rozšíření o takové druhy formátů by nebylo komplikovanější, než při tvorbě dosavadního výstupu (PHP) a je možné tento bod zařadit do možných inovačních návrhů pro budoucí práce v této problematice a tímto konkrétním generátorem.

Naproti tomu takové nástroje pro generování sestav jsou většinou budovány jako aplikace pro stolní počítače (tzv. desktop aplikace), jenž pak bývají závislé na platformě stanice (operační systém, doplňkové servisní balíčky apod.). Naproti tomu generátory sestav analyzované a implementované jak mnou, tak mým předchůdcem, jsou budovány jako internetový (tzv. webový) projekt, jenž je naopak na platformě nezávislé, v mém případě pak nezávislý také na nutnosti existence ovladačů ODBC a k využívání aplikace na straně klienta pak stačí pro elementární práci pouze jeden z předních internetových prohlížečů (viz kapitola Návrh implementace) a přístupnost k serverové stanici. Jednotlivé nástroje jsou uvedeny kapitole Reportovací služby, konkrétní příklad nad vybraným z nich pak ke konci v kapitole Implementace.

10 Závěr

Diplomové práce můžeme obecně v současnosti v oblasti informačních technologiích na naší fakultě elektrotechniky a informatiky rozdělit na dvě hlavní skupiny.

- Vytvoření nového projektu,
- rozšíření jiné diplomové práce,
- odborná stáž,

přičemž první dvě jmenované se obvykle překrývají a kombinují. Moje práce spadá v rámci této klasifikace velkou měrou do druhé skupiny, kdy jsem rozšířil práci [1], a zároveň nabídnul vlastní řešení hlavní části, kterou je implementace generátoru sestav.

Samotné implementaci předcházela řada analýz, kterou je možno zhlédnout buď formou příloh v elektronické podobě na přiloženém médiu, nebo formou výtahů přímo v textu této vytištěné práce v předchozích kapitolách.

Vedle těchto analýz, které vyplývají z vývoje informačního systému a jejich základní struktura a metodologie je předem definovaná, jsem práci po teoretické části zasadil do kontextu tzv. Bussiness Ingeligence, jejichž základy jsou v jedné z úvodních kapitol této práce spolu s uvedením nejvyužívanějších a nejznámějších nástrojů pro práci s nimi.

Generátor výstupních sestav, který jsem posléze implementoval, očekává na svém vstupu vždy jeden ze tři typů databáze, mezi které jsem, po konzultacích s vedoucí této diplomové práce, zařadil následující výčet.

- PostgreSQL,
- SQLite,
- Firebird.

Po stránce rozšíření a principu je toto pravděpodobně nejvýraznější rozdíl oproti původní práci minulého diplomanta. Jednou z dalších odlišností je také samotný přístup k datům, ať už na svém vstupu, či zejména výstupu generátoru, kde nevyužívám spojeních prostřednictvím jednotného ovladače ODBC, jako můj předchůdce, ale pomocí sady tzv. datových oficiálních prostředníků od distributora daného typu databáze ke konkrétní platformě (provider) na vstupu, resp. využití PDO ve výsledném vygenerovaném skriptu.

Tento přístup k datům pak s sebou nese výhody i nevýhody. Mezi ty kladné patří nezávislost na konfiguraci operačního systému takové pracovní stanice, ze které je generátor spouštěn (server), mezi nevýhody pak nutnost obstarat zmíněné datové prostředníky pro všechny typy databází, se kterými aplikace hodlá pracovat. Protože se ale tato nevýhoda vykazuje především pouze jednorázově ve fázi implementaci generátoru a na následný chod a tvorbu výstupních sestav a případně skriptů nemá vliv, byl zvolen právě tento přístup.

K přístupu k datům jsem se vrátil ještě v implementaci vlastního převodníku z jednoho typu dat (MySQL) na jiný (MSSQL). Protože se ale nejedná o primární úkol ani cíl této práce, nebyla mu věnována, zejména po analytické části, taková pozornost. V textu této práce jsem pak seznámil čtenáře této práce se základním principem a výsledkem testování na konkrétním příkladě, který proběhl

úspěšně. Celý projekt je pak rovněž přiložen v příloze. Zásadní nevýhodou této aplikace bude jeho postupná neaktuálnost, neboť pracuje nad konkrétním a přesně definovaným formátem vstupních dat vygenerovaných aplikací s názvem PHPMyAdmin, která se neustálé a dynamicky vyvíjí.

Implementační jazyk hlavní části práce, tj. generátoru sestav, zůstal shodný s jazykem práce, ze kterého je ta moje odvozována [1], jímž je C# (C Sharp). Ovšem v rámci rozšíření jsem zavedl a implementoval SW architektonický vzor MVC, jenž se, zejména v aplikacích pro technologie typu www, hojně využívá, a to zejména od konce 90. let, ačkoliv jeho základní principy a implementace jsou v oblasti IT známy a definovány již od roku 1979. Implementace takového vzoru, na platformě .NET označeného jako MVC 3, s sebou nese poměrně snadnou orientaci ve struktuře zdrojových souborů a tedy snadného rozšíření pro člověka znalého tohoto návrhového vzoru. V současnosti je na platformě .NET vydán MVC 4, v době zadání této diplomové práce to pak byl MVC 2, kde byly zavedeny tzv. oblasti (areas), což jsou ucelené oddělené skupiny, z nichž každá má své vlastní prvky MVC a v mé implementaci jsou využity (skupina pro uživatele, sestavy, spojení, apod.). V mé verzi aplikace je rovněž využita novinka dostupná teprve ve verzi MVC 3, kterou je nový systém pohledů (view-engine), jenž má za úkol zpřehlednit výsledný pohled (šablona) se značkami HTML i v případě existence většího množství exekutivního kódu (v jazyce C# nebo VB.NET) , než klasické stránky aspx.

Se změnou návrhového vzoru přišla také změna GUI, jenž je patrné zejména při spuštění průvodce sestavy či detailu již exitující sestavy. Rozhraní se snaží uživateli přinést podobný komfort, jaký mu mohou nabídnout desktopové aplikace různých analytických nástrojů, ale na rozdíl od nich se pak děje prostřednictvím (X)HTML a knihovny JQuery (zobrazení tabulek a atributů, jejich pořadí, závorkování podmínek a pod.) [13].

Funkční část generátoru přinesla rozšíření jak v uživatelské sekci, tak zejména při správě sestavy. Mezi hlavní, které můj generátor obsahuje, patří možnost v průvodci sestavy vytváření, resp. odstraňování tzv. poddotazů, a to bez omezení úrovně jejich zanoření.

Uživatelská sekce byla rozšířena o správu rolí, resp. vytváření nových a asociace s existujícími sestavami (výstupy) pro omezení přístupu k jejich detailu, tj. k veškerým informacím včetně možnosti vygenerování a stažení výsledného skriptu (v jazyce PHP). Zavedeno bylo také zaznamování význačných událostí (tzv. log), a to zejména pro operace vkládání, mazání a úprav pro zpětné dohledání změn.

Vzhledem k rozsáhlosti studované problematiky je možné na tuto práci navazovat a dále zpracovávat. Jako příklady možností její rozšíření či úprav jsem uvedl následující výčet.

- Přidání nových typů databází,
- sloučení typů databází z původní odvozené práce (MySQL, MSSQL, Oracle),
- výkonnostní (zátěžové) testování,
- rozšíření práce s výslednou vygenerovanou sestavou (např. vzhled, formátování, distribuce apod.).

11 Literatura

[1] **Bartoníček, Adam.** *Generátor výstupních sestav.* Ostrava : VŠB - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, 2010.

[2] **Lacko, Luboslav.** *Business Inteligence v SQL Serveru 2008.* Brno : Computer Press, a.s., 2009. ISBN 978-80-251-2887-9.

[3] **Jaša, Petr.** Datové sklady. [Online] Fakulta informačních technologií VUT. [Cited: Leden 3, 2012.] http://www.common.cz/attachments/118_petr_jasa_datove_sklady.pdf

[4] **Microsoft.** Podpora nástroje Crystal Reports pro Visual Studio. *Pomoc a podpora Microsoft.* [Online] [Cited: Březen 15, 2012.] http://support.microsoft.com/kb/317789/cs

[5] SQLite Úvodní oficiální stránja projektu. *SQLite.* [Online] Dwayne Richard Hipp. [Cited: Duben 29, 2012.] http://www.sqlite.org/

[6] **Group, PostgreSQL Global Development.** Úvodní oficiální stránka projektu. *PostgreSQL.* [Online] PostgreSQL Global Development Group. [Cited: Duben 29, 2012.] http://www.postgresql.org/

[7] **Šarmanová, Jana.** *Databázové a informační systémy.* Ostrava : VŠB - Technická univerzita Ostrava, 2007. ISBN 978-80-248-1499-5.

[8] —. *ISBN 978-80-248-1499-5.* Ostrava : VŠB – Technická univerzita Ostrava, 2007. ISBN 978-80-248-1500-8.

[9] —. *Teorie zpracování dat.* Ostrava : VŠB - Technická univerzita Ostrava, 2007. ISBN 978-80-248-1498-8.

[10] Entity-relationship model. *Wikipedia, otevřená encyklopedie.* [Online] [Cited: Duben 13, 2012.] http://cs.wikipedia.org/wiki/Entity-relationship_model

[11] **C., Martin Robert.** *Čistý kód.* Brno : Computer Press, a.s., 2009. ISBN 978-80-251-2285-3.

[12] **Conery, Rob, et al.** *Professional ASP.NET MVC 1.0.* Crosspoint Boulevard, Indianapolis : Wiley Publishing, Inc., 2009. ISBN: 978-0-470-38461-9.

[13] JQuery API Browser. [Online] [Cited: Duben 27, 2012.] http://jquery.bassistance.de/api-browser-1.2/

[14] **Jiří, Kosek.** *PHP tvorba interaktivních internetových aplikací.* Praha : Grada Publishing, 1999. ISBN 80-7169-373-1.

[15] Úvodní oficiální stránka projektu. *Wamp Server.* [Online] [Cited: Duben 29, 2012.] http://www.wampserver.com/en/

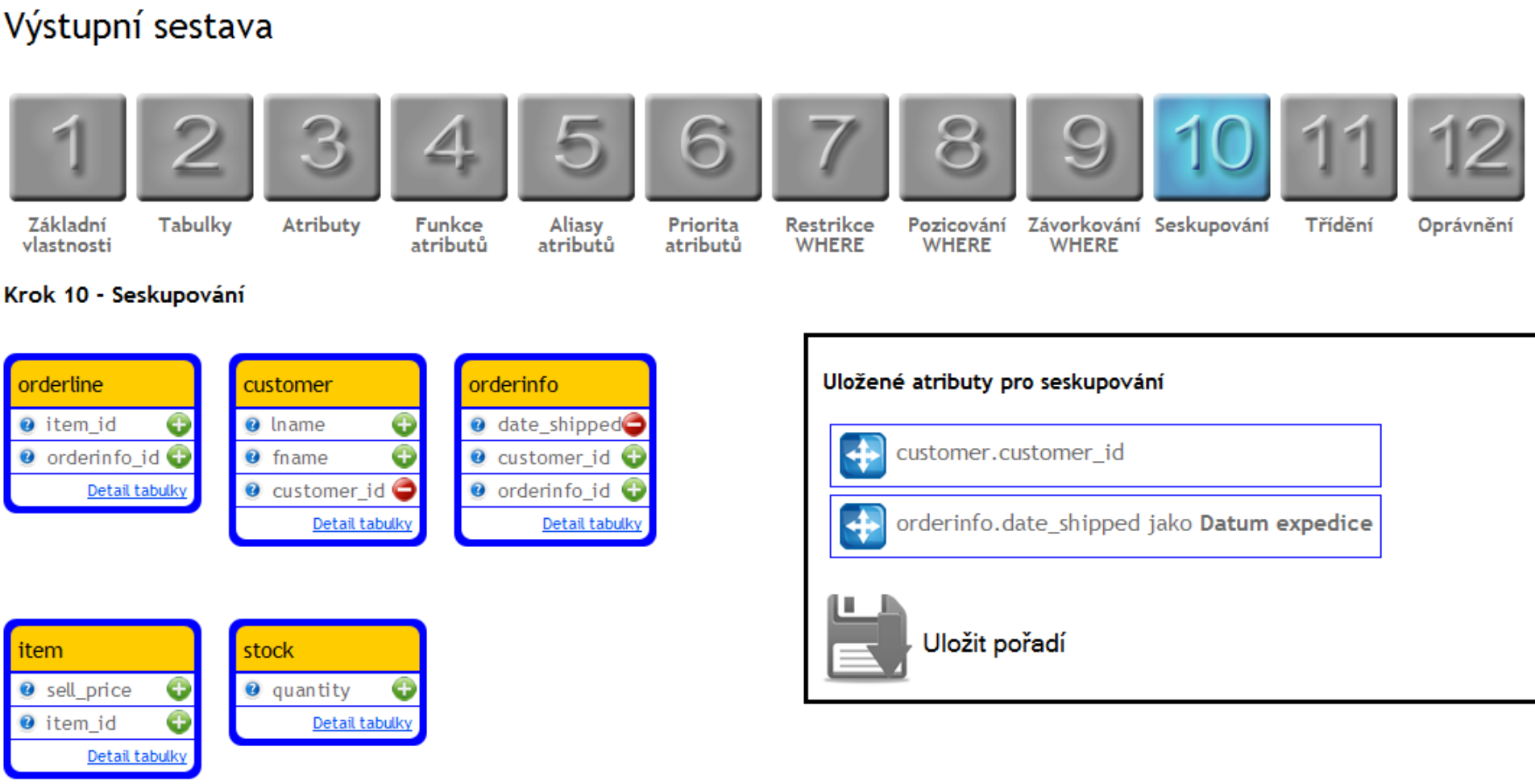
[16] JQuery. [Online] [Cited: duben 28, 2012.] http://www.jquery.com

[17] Úvodní oficiální stránka projektu Firebird. *Firebird.* [Online] Firebird Foundation. [Cited: Duben 29, 2012.] http://www.firebirdsql.org/

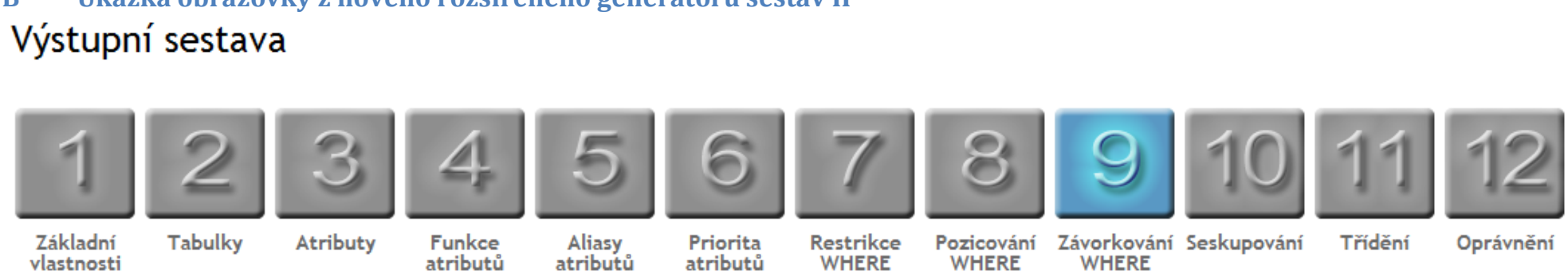
12 Seznam příloh

A Ukázka obrazovky z nového rozšířeného generátoru sestav I.
B Ukázka obrazovky z nového rozšířeného generátoru sestav II.
C Návrh GUI pro doplňkovou aplikaci - konvertoru dat.
D Adresářová struktura DVD.

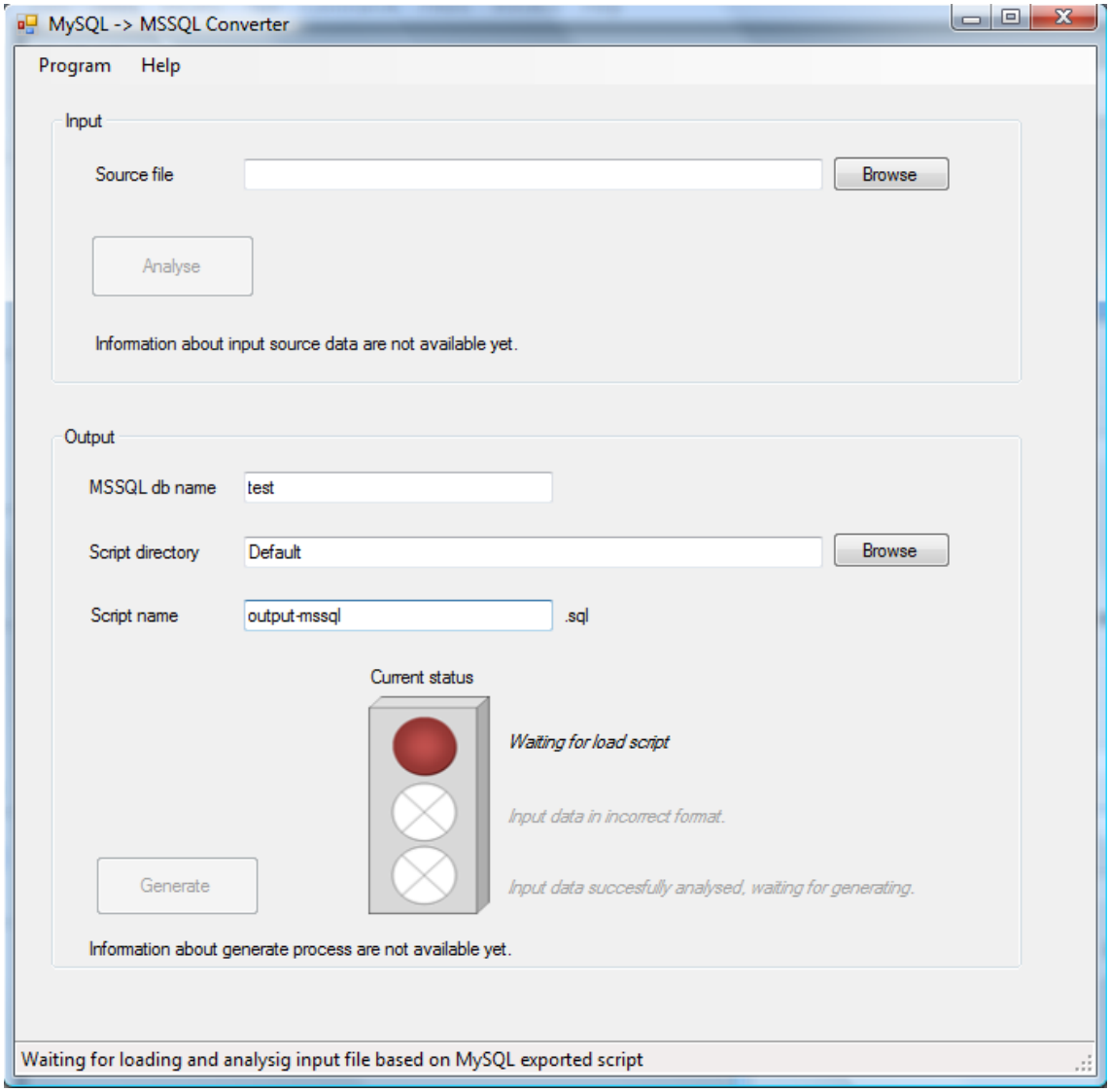
A Ukázka obrazovky z nového rozšířeného generátoru sestav I



B Ukázka obrazovky z nového rozšířeného generátoru sestav II



C Návrh GUI pro doplňkovou aplikaci - konvertoru dat



D Adresářová struktura DVD

/Generator/Project	Rozšířený generátor výstupních sestav.
/Generator/Input/	Konkrétní testovací vstupní data (databáze).
/Generator/Output	Vygenerované výstupní data (skripty).
/Generator/Testing	Testování rozšířeného generátoru.
/Attachments	Další přílohy.
/Attachments/User's Guide	Uživatelská příručka rozšířeného generátoru (PDF).
/Attachments/Convertor	Vlastní transformátor (konvertor) dvou typů databází.
/Attachments/Convertor/Project	Samotný projekt konvertoru.
/Attachments/Convertor/Input	Testovací vstupní data (databáze).
/Attachments/Convertor/Output	Výsledné výstupní data (databáze).
/Attachments/Convertor/Diagrams	Doprovodné diagramy pro konvertor.
/Attachments/Data Analysis	Datová analýza rozšířeného generátoru.
/Attachments/Programmer's Guide	Programátorská příručka rozšířeného generátoru (HTML/TEX).
/Attachments/Function Analysis	Funkční analýza rozšířeného generátoru (PDF).
/Attachments/Other Tools Reports	Report vytvořený v prostředí MS SQL Server 2008
/Programs	Aplikace pro podporu rozšířeného generátoru.
/Programs/SQLite	SQLite.
/Programs/Firebird	Firebird.
/Programs/PostgreSQL	PostgreSQL.
/Programs/Wamp Server	Server obsahující Apache a MySQL (32/64b).
/Text	Textová část diplomové práce (Vypracování).